ALGORITHMS AND ANALYSIS FOR NON-CONVEX OPTIMIZATION PROBLEMS IN MACHINE LEARNING

A Dissertation
Presented to
The Academic Faculty

By

Bo Xie

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science & Engineering

Georgia Institute of Technology

August 2017

ALGORITHMS AND ANALYSIS FOR NON-CONVEX OPTIMIZATION PROBLEMS IN MACHINE LEARNING

Approved by:

Professor Le Song, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Professor Santosh Vempala
School of Computer Science
Georgia Institute of Technology

Professor Hongyuan Zha
School of Computational Science and Engineering
Georgia Institute of Technology

Professor Byron Boots
School of Interactive Computing
Georgia Institute of Technology

Professor Anima Anandkumar
School of Information and Computer Sciences
University of California, Irvine

Date Approved: May 1, 2017

To my parents, Shuyu and Emily

ACKNOWLEDGEMENTS

Many people have contributed to the completion of this PhD thesis. I deeply appreciate their support and the opportunities they have offered me. I am fortunate enough to have worked with some of the best researchers in my field, and I am grateful to them all: Anima Anandakumar, Nina Balcan, Byron Boots, Polo Chau, Douglas Eck, Xin Gao, Sanjiv Kumar, Santosh Vempala and David Woodruff. I am also thankful for the support from my collaborators: Bo Dai, Nan Du, Mehrdad Farajtabar, Niao He, Kenji Kawaguchi, Yingyu Liang and Yichen Wang.

I would like to thank my fellow students for the constructive discussions and time. They have both helped me grow academically and supported me in life. I am grateful to: Bo Dai, Hanjun Dai, Nan Du, Mehrdad Farajtabar, Niao He, Yangfeng Ji, Elias Khalil, Da Kuang, Liangda Li, Shuang Li, Yingyu Liang, Weiyang Liu, Yongchao Liu, Rakshit Trivedi, Yichen Wang, Yanming Zhang and Yuyu Zhang.

I would like to express my gratitude to Dacheng Tao, Xiaogang Wang, Parag Chordia, and Polo Chau for their support in my early days of PhD studies. I am also grateful to my committee members, Santosh Vempala, Hongyuan Zha, Byron Boots and Anima Anandkumar for the valuable feedbacks.

I feel strongly indebted to my thesis advisor Professor Le Song. Without his visionary guidance and the opportunities he has provided me, I could not achieve what I have now.

Finally, I owe tremendously to the love and support of my family. You are always the most important part of my life and I love you all so much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, we propose efficient algorithms and provide theoretical analysis through the angle of spectral methods for some important non-convex optimization problems in machine learning. Specifically, we focus on two types of non-convex optimization problems: learning the parameters of latent variable models and learning in deep neural networks.

Part I. Spectral methods for latent variable estimation

Learning latent variable models is traditionally framed as a non-convex optimization problem through Maximum Likelihood Estimation (MLE). For some specific models such as multi-view model, we can bypass the non-convexity by leveraging the special model structure and convert the problem into spectral decomposition through Methods of Moments (MM) estimator. In this thesis, we propose a novel algorithm that can flexibly learn a multi-view model in a non-parametric fashion. It estimates the conditional distributions of the latent variable model by decomposing operators in a functional space.

We then demonstrate one application of spectral methods in the task of DNA motif prediction. By modeling the DNA sequence as a HMM and learn the representation with spectral methods, we can efficiently compute the posterior distribution of the latent variables without being trapped in local optima. We then feed these latent features into a classifier and achieve superior performance than that obtained by hand-crafted features.

Part II. Scalable algorithms to solve nonlinear spectral methods

One obstacle of applying the nonparametric spectral methods to large datasets is that it scales at least quadratically with the number of data points. To overcome the issue, we propose two versions of scalable nonlinear spectral algorithms. One version, called doubly stochastic gradient descent, uses sampling to approximate two expectations in the problem, and it achieves better balance of computation and statistics by adaptively growing the model as more data arrive. Although it is still a non-convex optimization problem, the algorithm is guaranteed to converge at the rate of $O(1/t)$ where t is the number of iterations.

Another version is the distributed kernel principle component analysis (KPCA) algorithm. The proposed algorithm estimates leverage scores to only sample a few representative data points from the whole dataset. To reduce communication overhead, the leverage scores are approximated by sketched data points. By carefully controlling the balance between communication and approximation error, we obtain a distributed algorithm that is nearly optimally communication efficient.

Part III. Analysis of neural network learning

Learning with neural networks is a difficult non-convex problem while simple gradient-based methods achieve great success in practice. In this part of the thesis, we try to understand the optimization landscape of learning one-hidden-layer networks with Rectified Linear (ReLU) activation functions. By directly analyzing the structure of the gradient, we can show neural networks with diverse weights have no spurious local optima. This partly explains the empirical success of gradient descent since a stationary point leads to a global optimum under diversity conditions on the neural weights.

Inspired by the analysis, we introduce semi-random units, which sit between fully adjustable ReLU units and random features. Semi-random units possess nice theoretical properties despite the non-convex nature of the optimization problem. In particular, networks with such units have no spurious local optima and gradient descent converges to the global optimum. Moreover, semi-random features only use slightly more units to reach comparable performance as ReLU and they use much fewer units compared with random features.

CHAPTER 1
INTRODUCTION

Most machine learning problems can be formulated as optimization problems. For example, the most commonly used criterion is the Maximum Likelihood Estimation (MLE) where one maximizes the log-likelihood of the data given a particular model. In addition, most supervised learning algorithms can fit under the empirical risk minimization (ERM) framework where one minimizes the loss averaged over the training set plus a regularization term. Even from the Bayesian perspective, variational Bayesian methods perform inference by solving an optimization problem.

One particular class of optimization problems, namely convex optimization, has traditionally received much attention from the machine learning community due to its nice theoretical properties and existence of efficient algorithms. For convex problems, any local optimum is a global optimum, so one can use simple local search algorithms such as gradient descent to reach the optima with guarantees. These guarantees are so powerful that gradient descent and its variants are the standard approach to solve large-scale machine learning problems.

However, many important problems cannot be formulated as convex optimization or will be more computationally expensive than their non-convex counterparts. For instance, latent variable models are a large family of probabilistic graphical models that involve non-convex optimization. In addition, matrix completion, another popular task, can be more efficiently solved through alternating minimization than the convex nuclear norm regularization. Moreover, deep learning is filled with highly complex and difficult non-convex optimization problems.

There are mainly two reasons for why non-convex optimization is necessary for machine learning. The first one is that in many cases data exhibit "cluster" structures. In order

to capture such structures, a model will likely consists of components that correspond to such "clusters" and this usually leads to non-convex optimization such as in k-means and matrix decomposition. Another reason is that non-convex formulations are usually more efficient in terms of number of parameters to succinctly represent a joint distribution or a function class. Examples include latent graphical models and deep neural networks.

Unfortunately, non-convex formulation achieves more efficient representation at the expense of intractable optimization. In general, no efficient algorithms exist for non-convex problems since there may be numerous local optima or saddle points. One can still use techniques such as gradient descent in practice but there are no guarantees for such procedures. In fact, how to optimize correctly is a fundamental research question in learning neural networks. One needs to know many tricks of the trade and it is more an art than a science to successfully train a deep network.

In this thesis, we tackle some specific non-convex problems from the angle of spectral methods and analysis. In the case of latent variable models, we bypass the non-convexity by resorting to the Methods of Moments (MM) estimators to solve a set of nonlinear equations. Such equations can be efficiently solved by spectral decomposition because these latent variable models exhibit symmetric low-rank tensor structure. In the case of analyzing convergence properties of deep neural networks, we can express the gradient as the product of a special matrix and the residual vector. By lower bounding the spectrum of the special matrix, we can show a stationary point is a global optimum.

Computationally, we have also proposed novel algorithms to solve large-scale non-convex optimization problems. In particular, we focus on kernel PCA, which is a basic computation unit used by many other algorithms. We have proposed both stochastic and distributed versions. The stochastic algorithm is called doubly stochastic gradient, which uses two sources of randomness to speed up computation. The distributed version leverages sketches to reduce communication overheads.

1.1 Thesis structure

This thesis is organized into three parts. In the first part, we propose spectral methods for learning latent variable models. Chapter 2 introduces a non-parametric algorithm for estimating the conditional distributions of a multi-view model. In Chapter 3, we showcase one application of spectral methods in bioinformatics.

The second part centers around tackling computational issues of spectral algorithms in a functional space. The main computation boils down to kernel PCA. In the next two chapters, we introduce two versions of scalable algorithms for kernel PCA. The first one is a stochastic optimization algorithm, called doubly stochastic gradient, that uses a few samples to approximate an expectation. In Chapter 4, we introduce this algorithm in the non-convex setting and prove that it is guaranteed to converge to the global optima. The second version of scalable algorithm is distributed kernel PCA, which we illustrate in Chapter 5.

In the last part, we study the non-convex problem associated with learning neural networks. Recently, deep learning has been extremely successful with simple gradient descent even on highly non-convex objectives. In Chapter 6, we try to understand this phenomenon by analyzing a one-hidden-layer neural network and show under some conditions there are no spurious local optima. Chapter 7 improves the analysis by introducing deep semi-random features that relax the condition on the weights which may be difficult to guarantee during gradient descent.

1.2 Thesis contribution

The thesis include both theoretical and algorithmic contributions to the field of machine learning. These contributions are:

1. A principled algorithm to solve some specific latent variable models without being trapped in local optima.

2. Scalable algorithms to solve a key non-convex optimization problem.

3. Theoretical analysis of the optimization landscape of learning neural networks.

The set of publications related to this thesis are listed below:

1. K. Kawaguchi, B. Xie, L. Song. Deep Semi-Random Features for Nonlinear Function Approximation. (submitted to ICML, 2017)

2. B. Xie, Y. Liang, and L. Song. Diversity Leads to Generalization in Neural Networks. In: AI & Statistics (AISTATS), 2017.

3. M. Balcan, Y. Liang, L. Song, D. Woodruff, and B. Xie. Distributed Kernel Principal Component Analysis. In: Knowledge Discovery and Data Mining (KDD), 2016.

4. Y. Wang, B. Xie, N. Du, and L. Song. Isotonic Hawkes Processes. In: International Conference on Machine Learning (ICML), 2016.

5. B. Xie, Y. Liang, and L. Song. Scale Up Nonlinear Component Analysis with Doubly Stochastic Gradients. In: Neural Information Processing Systems (NIPS), 2015.

6. A. Shaban, M. Farajtabar, B. Xie, L. Song and B. Boots. Learning Latent Variable Models by Improving Spectral Solutions with Exterior Point Method. In: Uncertainty in Artificial Intelligence (UAI), 2015.

7. B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M. Balcan and L. Song. Scalable Kernel Methods via Doubly Stochastic Gradients. In: Neural Information Processing Systems (NIPS), 2014.

8. L. Song, A. Anamdakumar, B. Dai and B. Xie. Nonparametric Estimation of Multi-View Latent Variable Models. In: International Conference on Machine Learning (ICML), 2014.

9. B. Xie, B. Jankovic, V. Bajic, L. Song and X. Gao. PolyA motif prediction using spectral latent features from human DNA sequences. In: Annual International Conference on Intelligent Systems for Molecular Biology (ISMB), 2013.

PART I: SPECTRAL METHODS FOR LATENT VARIABLE MODELS

Latent variable models are a powerful class of probabilistic graphical models that decompose the joint distribution into a mixture of factorized distributions. Such decomposition not only provides a more efficient representation but also captures the corresponding structure within data. Some prime examples are Gaussian Mixture Models (GMM) [1], Hidden Markov Models (HMM) [2] and Latent Dirichlet Allocation (LDA) [3].

One pressing problem in learning these latent variable models through Maximum Likelihood Estimation (MLE) is that the optimization problem is non-convex. The most popular algorithm to learn these models is the Expectation Maximization (EM) [4] algorithm, which is a local search heuristic and the solution quality highly depends on the initialization.

In this part, we discuss Methods of Moments (MM) as an alternative to MLE to learn the latent variable models. Unlike MLE, MM estimates the parameters by solving a set of equations where the empirical moments are equal to the population moments. For some specific models, the population moments have a nice tractable form which allows one to solve them through spectral decompositions. This approach is called "spectral methods" and it has seen many successful applications.

In Chapter 2, we introduce a non-parametric algorithm for estimating conditional distributions of a multi-view model. Previous approaches can only handle discrete distributions or parametric forms. The non-parametric estimation technique can model many more complex distributions without assuming prior knowledge of the specific form.

In Chapter 3, we demonstrate the effectiveness of spectral algorithms in the application of bioinformatics. In particular, these features learned through the spectral algorithms achieve superior performance than that of hand-crafted features.

CHAPTER 2
NONPARAMETRIC ESTIMATION OF MULTI-VIEW LATENT VARIABLE MODELS

Spectral methods have greatly advanced the estimation of latent variable models, generating a sequence of novel and efficient algorithms with strong theoretical guarantees. However, current spectral algorithms are largely restricted to mixtures of discrete or Gaussian distributions. In this chapter, we propose a kernel method for learning multi-view latent variable models, allowing each mixture component to be nonparametric and learned from data in an *unsupervised* fashion. The key idea of our method is to embed the joint distribution of a multi-view latent variable model into a reproducing kernel Hilbert space, and then the latent parameters are recovered using a robust tensor power method. We establish that the sample complexity for the proposed method is quadratic in the number of latent components and is a low order polynomial in the other relevant parameters. Thus, our nonparametric tensor approach to learning latent variable models enjoys good sample and computational efficiencies. As a special case of our framework, we also obtain a first *unsupervised* conditional density estimator of the kind with provable guarantees. In both synthetic and real world datasets, the nonparametric tensor power method compares favorably to EM algorithm and other spectral algorithms.

2.1 Introduction

Recently, there is a surge of interest in designing spectral algorithms for estimating the parameters of latent variable models [5, 6, 7, 8, 9, 10, 11]. Compared to the Expectation-Maximization (EM) algorithm [4] traditionally used for this task, spectral algorithms are better in terms of their computational efficiency and provable guarantees. However, current spectral algorithms are largely restricted to mixture of discrete or Gaussian distributions,

e.g. [10, 12]. When the mixture components are distributions other than these standard distributions, the theoretical guarantees for these algorithms are no longer applicable, and their empirical performance can be very poor.

We propose a kernel method for obtaining sufficient statistics of a multi-view latent variable model (for $\ell \geq 3$),

$$\mathbb{P}\left(\{X_t\}_{t\in[\ell]}\right) = \sum\nolimits_{h\in[k]} \mathbb{P}(h) \cdot \prod\nolimits_{t\in[\ell]} \mathbb{P}(X_t|h), \tag{2.1}$$

given samples only from the observed variables $\{X_t\}_{t\in[\ell]}$, but *not* the hidden variable H. These statistics allow us to answer integral query, $\int_{\mathcal{X}} f(x_t)\, d\mathbb{P}(x_t|h)$, for functions f from a reproducing kernel Hilbert space (RKHS) *without* the need to assume any parametric form for the involved latent component $\mathbb{P}(X_t|h)$ (we call this setting "*unsupervised*"). Note that this is a very challenging problem, since we do not have samples to directly estimate $\mathbb{P}(X_t|h)$. Hence traditional kernel density estimator does not apply. Furthermore, the non-parametric form of $\mathbb{P}(X_t|h)$ renders previous spectral methods inapplicable.

Our solution is to embed the distribution of the observed variables in such a model into a reproducing kernel Hilbert space, and exploit tensor decomposition of the embedded distribution (or covariance operators) to recover the unobserved embedding $\mu_{X_t|h} = \int_{\mathcal{X}} \phi(x)\, d\mathbb{P}(x|h)$ of the mixture components. The key computation of our algorithm involves a kernel singular value decomposition of the two-view covariance operator, followed by a robust tensor power method on the three-view covariance operator. These standard matrix operations makes the algorithm very efficient and easy to deploy.

Although kernel methods have been previously applied to learning latent variable models, none of them can provably recover the exact latent component $\mathbb{P}(X_t|h)$ or its sufficient statitiscs to support integral query on this distribution. For instance, [6, 8, 13] estimated an (unknown) invertible transformation of the sufficient statistics of the latent component $\mathbb{P}(X_t|h)$, and only supported integral query associated with the distribution of the observed

variables. [14] used kernel independence measure to cluster data points, and treated each cluster as a latent component. Besides computational issues, it is also difficult to provide theoretical guarantees to such an approach since the clustering step only finds a local minimum. [15] designed an EM-like algorithm for learning the conditional densities in latent variable models. This algorithm alternates between the E-step, proportional assignment of data points to components, and the M-step, kernel density estimation based on weighted data points. Similarly, theoretical analysis of such a local search heurstic is difficult.

The kernel algorithm proposed in this chapter is also significantly more general than the previous spectral algorithms which work only for distributions with parametric assumptions [10, 12]. In fact, when we use the delta kernel, our algorithm recovers the previous algorithm of [10] for discrete mixture components as a special case. When we use universal kernels, such as the Gaussian RBF kernel, our algorithm can recover Gaussian mixture components as well as mixture components with other distributions. In this sense, our work also provides a unifying framework for previous spectral algorithms. We prove sample complexity bounds for the nonparametric tensor power method and show that it is both computational and sample efficient. As a special case of our framework, we also obtain a first *unsupervised* conditional density estimator of the kind with provable guarantees. Furthermore, our approach can also be generalized to other latent variable learning tasks such as independent component analysis and latent variable models with Dirichlet priors.

Experimentally, we corroborate our theoretical results by comparing our algorithm to the EM algorithm and previous spectral algorithms. We show that when the model assumptions are correct for the EM algorithm and previous spectral algorithms, our algorithm converges in terms of estimation error to these competitors. In the opposite cases when the model assumptions are incorrect, our algorithm is able to adapt to the nonparametric mixture components and beating alternatives by a very large margin.

2.2 Preliminary

We denote by X a random variable with domain $\mathcal{X}$, and refer to instantiations of X by the lower case character, x. We endow $\mathcal{X}$ with some σ-algebra $\mathscr{A}$ and denote a distributions (with respect to $\mathscr{A}$) on $\mathcal{X}$ by $\mathbb{P}(X)$. For the multi-view model in equation (2.1), we also deal with multiple random variables, $X_1, X_2, \ldots, X_\ell$, with joint distribution $\mathbb{P}(X_1, X_2, \ldots, X_\ell)$. For simplicity of notation, we assume that the domains of all $X_t, t \in [\ell]$ are the same, but the methodology applies to the cases where they have different domains. Furthermore, we denote by H a hidden variable with domain $\mathcal{H}$ and distribution $\mathbb{P}(H)$.

A *reproducing kernel Hilbert space (RKHS)* $\mathcal{F}$ on $\mathcal{X}$ with a kernel $\kappa(x, x')$ is a Hilbert space of functions $f(\cdot) : \mathcal{X} \mapsto \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. Its element $\kappa(x, \cdot)$ satisfies the reproducing property: $\langle f(\cdot), \kappa(x, \cdot) \rangle_{\mathcal{F}} = f(x)$, and consequently, $\langle \kappa(x, \cdot), \kappa(x', \cdot) \rangle_{\mathcal{F}} = \kappa(x, x')$, meaning that we can view the evaluation of a function f at any point $x \in \mathcal{X}$ as an inner product. Alternatively, $\kappa(x, \cdot)$ can be viewed as an implicit feature map $\phi(x)$ where $\kappa(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$. In this chapter, we will focus on $\mathcal{X} = \mathbb{R}^d$, and the *normalized* Gaussian RBF kernel

$$\kappa(x, x') = \exp(-\|x - x'\|^2 / (2s^2)) / (\sqrt{2\pi} s^d). \tag{2.2}$$

But kernel functions have also been defined on graphs, time series, dynamical systems, images and other structured objects [16]. Thus the methodology presented below can be readily generalized to a diverse range of data types as long as kernel functions are defined.

2.3 Kernel Embedding of Distributions

Kernel embeddings of distributions are *implicit* mappings of distributions into potentially *infinite* dimensional RKHS. The kernel embedding approach represents a distribution by

an element in the RKHS associated with a kernel function [17],

$$\mu_X := \mathbb{E}_X [\phi(X)] = \int_{\mathcal{X}} \phi(x) \, d\mathbb{P}(x), \tag{2.3}$$

where the distribution is mapped to its expected feature map, *i.e.*, to a point in a potentially infinite-dimensional and implicit feature space. By the reproducing property of an RKHS, the kernel embedding is a sufficient statistic for integral query $\forall f \in \mathcal{F}$, *i.e.*, $\int_{\mathcal{X}} f(x) \, d\mathbb{P}(x) = \langle \mu_X, f \rangle_{\mathcal{F}}$. Kernel embedding of distributions has rich representational power. The mapping is injective for characteristic kernels [18]. That is, if two distributions, $\mathbb{P}(X)$ and $\mathbb{Q}(X)$, are different, they are mapped to two distinct points in the RKHS. For domain $\mathbb{R}^d$, many commonly used kernels are characteristic, such as the normalized Gaussian RBF kernel.

Kernel embeddings can be readily generalized to joint distributions of two or more variables using tensor product feature maps. We can embed the joint distribution of two variables X_1 and X_2 into a tensor product feature space $\mathcal{F} \times \mathcal{F}$ by $\mathcal{C}_{X_1 X_2} := \int_{\mathcal{X} \times \mathcal{X}} \phi(x_1) \otimes \phi(x_2) \, d\mathbb{P}(x_1, x_2)$, where the reproducing kernel for the tensor product features satisfies $\langle \phi(x_1) \otimes \phi(x_2), \phi(x_1') \otimes \phi(x_2') \rangle_{\mathcal{F} \times \mathcal{F}} = \kappa(x_1, x_1') \, \kappa(x_2, x_2')$. By analogy, we can also define $\mathcal{C}_{X_1 X_2 X_3} := \mathbb{E}_{X_1 X_2 X_3}[\phi(X_1) \otimes \phi(X_2) \otimes \phi(X_3)]$.

Given a sample $\mathcal{D}_X = \{x^1, \ldots, x^m\}$ of size m drawn *i.i.d.* from $\mathbb{P}(X)$, the empirical kernel embedding can be estimated simply as $\widehat{\mu}_X = \frac{1}{m} \sum_{i=1}^{m} \phi(x^i)$ with an error $\| \widehat{\mu}_X - \mu_X \|_{\mathcal{F}}$ scaling as $O_p(m^{-\frac{1}{2}})$ [17]. Similarly, $\mathcal{C}_{X_1 X_2}$ and $\mathcal{C}_{X_1 X_2 X_3}$ can be estimated as $\widehat{\mathcal{C}}_{X_1 X_2} = \frac{1}{m} \sum_{i=1}^{m} \phi(x_1^i) \otimes \phi(x_2^i)$, and $\widehat{\mathcal{C}}_{X_{1:3}} = \frac{1}{m} \sum_{i=1}^{m} \phi(x_1^i) \otimes \phi(x_2^i) \otimes \phi(x_3^i)$ respectively. Note that we never explicitly compute the feature maps $\phi(x)$ for each data point. Instead, most of the computation required for subsequent statistical inference using kernel embeddings can be reduced to the Gram matrix manipulation.

2.3.1 Kernel Embedding as Multi-Linear Operator

The joint embeddings can also be viewed as an uncentered covariance operator $\mathcal{C}_{X_1X_2} : \mathcal{F} \mapsto \mathcal{F}$ by the standard equivalence between a tensor product feature and a linear map. That is, given two functions $f_1, f_2 \in \mathcal{F}$, their covariance can be computed by

$$\mathbb{E}_{X_1X_2}[f_1(X_1)f_2(X_2)] = \langle f_1, \mathcal{C}_{X_1X_2} f_2 \rangle_{\mathcal{F}},$$

or equivalently $\langle f_1 \otimes f_2, \mathcal{C}_{X_1X_2} \rangle_{\mathcal{F}\times\mathcal{F}}$, where in the former we view $\mathcal{C}_{XY}$ as an operator while in the latter we view it as an element in tensor product feature space. By analogy, $\mathcal{C}_{X_1X_2X_3}$ (with shorthand $\mathcal{C}_{X_{1:3}}$) can be regarded as a multi-linear operator from $\mathcal{F} \times \mathcal{F} \times \mathcal{F}$ to $\mathbb{R}$. It will be clear from the context whether we use $\mathcal{C}_{X_{1:3}}$ as an operator between two spaces or as an element from a tensor product feature space. For generic introduction to tensors, please see [19].

In the multi-linear operator view, the application of $\mathcal{C}_{X_{1:3}}$ to a set of elements $\{f_1, f_2, f_3 \in \mathcal{F}\}$ can be defined using the inner product from the tensor product feature space, ***i.e.***,

$$\mathcal{C}_{X_{1:3}} \times_1 f_1 \times_2 f_2 \times_3 f_3 := \langle \mathcal{C}_{X_{1:3}},\ f_1 \otimes f_2 \otimes f_3 \rangle_{\mathcal{F}^3}$$

which is further equal to $\mathbb{E}_{X_1X_2X_3}[\prod_{t\in[3]} \langle \phi(X_t),\ f_t \rangle_{\mathcal{F}}]$. Furthermore, we can define the Hilbert-Schmidt norm $\|\cdot\|$ as $\|\mathcal{C}_{X_{1:3}}\|^2 = \sum_{i_1,i_2,i_3=1}^{\infty} (\mathcal{C}_{X_{1:3}} \times_1 u_{i_1} \times_2 u_{i_2} \times_3 u_{i_3})^2$ using three collections of orthonormal bases $\{u_{i_1}\}_{i_1=1}^{\infty}$, $\{u_{i_2}\}_{i_2=1}^{\infty}$, and $\{u_{i_3}\}_{i_3=1}^{\infty}$.

The joint embedding, $\mathcal{C}_{X_1X_2}$, can be viewed as infinite dimensional matrices. For instance, we can perform singular value decomposition $\mathcal{C}_{X_1X_2} = \sum_{i=1}^{\infty} \sigma_i \cdot u_{i_1} \otimes u_{i_2}$, where $\sigma_i \in \mathbb{R}$ are singular values ordered in nonincreasing manner, and $\{u_{i_1}\}_{i_1=1}^{\infty} \subset \mathcal{F}, \{u_{i_2}\}_{i_2=1}^{\infty} \subset \mathcal{F}$ are singular vectors and orthonormal bases. The rank of $\mathcal{C}_{X_1X_2}$ is the smallest k such that $\sigma_i = 0$ for $i > k$.

(a) Naïve Bayes model (b) Hidden Markov model

Figure 2.1: Examples of multi-view latent variable models.

2.4 Multi-View Latent Variable Models

Multi-view latent variable models studied in this chapter are a special class of Bayesian networks in which (*i*) observed variables $X_1, X_2, \ldots, X_\ell$ are conditionally independent given a discrete latent variable H, and (*ii*) the conditional distributions, $\mathbb{P}(X_t|H)$, of the $X_t, t \in [\ell]$ given the hidden variable H can be different. The conditional independent structure of a multi-view latent variable model is illustrated in Figure 2.1(a), and many complicated graphical models, such as the hidden Markov model in Figure 2.1(b), can be reduced to a multi-view latent variable model. For simplicity of exposition, we will explain our method using the model with symmetric view. That is the conditional distribution are the same for each view, *i.e.*, $\mathbb{P}(X|h) = \mathbb{P}(X_1|h) = \mathbb{P}(X_2|h) = \mathbb{P}(X_3|h)$. In Appendix A.1, we will show that multi-view models with different views can be reduced to ones with symmetric view.

2.4.1 Conditional Embedding Operator

For simplicity of exposition, we focus on a simple model with three observed variables ($\ell = 3$). Suppose $H \in [k]$, then we can embed each conditional distribution $\mathbb{P}(X|h)$ corresponding to a particular value of $H = h$ as

$$\mu_{X|h} = \int_{\mathcal{X}} \phi(x)\, d\mathbb{P}(x|h). \tag{2.4}$$

If we vary the value of H, we obtain the kernel embedding for different $\mathbb{P}(X|h)$. Conceptually, we can tile these embeddings into a matrix (with infinite number of rows)

$$\mathcal{C}_{X|H} = \left(\mu_{X|h=1}, \mu_{X|h=2}, \ldots, \mu_{X|h=k}\right), \tag{2.5}$$

which is called the conditional embedding operator. If we use the standard basis e_h in $\mathbb{R}^k$ to represent each value of h, we can retrieve each $\mu_{X|h}$ from $\mathcal{C}_{X|H}$ by

$$\mu_{X|h} = \mathcal{C}_{X|H} e_h \tag{2.6}$$

Once we have the conditional embedding $\mu_{X|h}$, we can compute the conditional expectation of a function $f \in \mathcal{F}$ as $\int_{\mathcal{X}} f(x)\, d\mathbb{P}(x|h) = \left\langle f, \mu_{X|h} \right\rangle_{\mathcal{F}}$.

Remarks. For data from $\mathbb{R}^d$ and the normalized Gaussian RBF kernel in (2.2), the conditional density $p(x|h)$ exists, and it can be approximated by the embedding as $\widetilde{p}(x|h) := \left\langle \phi(x), \mu_{X|h} \right\rangle_{\mathcal{F}} = \mathbb{E}_{X|h}[\kappa(x, X)]$. Essentially, this is the convolution of the conditional density with the kernel function.

For continuous density $p(x|h)$ with suitable smoothness conditions, the approximation error is of the order [20]

$$|p(x|h) - \widetilde{p}(x|h)| = O(s^2). \tag{2.7}$$

2.4.2 Factorized Kernel Embedding

For multi-view latent variable models, $\mathbb{P}(X_1, X_2)$ and $\mathbb{P}(X_1, X_2, X_3)$, can be factorized respectively as

$$\mathbb{P}(x_1, x_2) = \sum\nolimits_{h \in [k]} \mathbb{P}(x_1|h)\, \mathbb{P}(x_2|h)\, \mathbb{P}(h), \text{ and}$$
$$\mathbb{P}(x_1, x_2, x_3) = \sum\nolimits_{h \in [k]} \mathbb{P}(x_1|h)\, \mathbb{P}(x_2|h)\, \mathbb{P}(x_3|h)\, \mathbb{P}(h).$$

Since we assume the hidden variable $H \in [k]$ is discrete, we let $\pi_h := \mathbb{P}(h)$. Furthermore, if we apply Kronecker delta kernel $\delta(h, h')$ with feature map e_h, then the embeddings for $\mathbb{P}(H)$

$$\mathcal{C}_{HH} = \mathbb{E}_H[e_H \otimes e_H] = \Big(\pi_h \delta(h, h') \Big)_{h,h' \in [k]}, \text{ and}$$

$$\begin{aligned} \mathcal{C}_{HHH} &= \mathbb{E}_H[e_H \otimes e_H \otimes e_H] \\ &= \Big(\pi_h \, \delta(h, h') \, \delta(h', h'') \Big)_{h,h',h'' \in [k]} \end{aligned}$$

are diagonal tensors. Making use of $\mathcal{C}_{HH}$ and $\mathcal{C}_{HHH}$, and the factorization of the distributions $\mathbb{P}(X_1, X_2)$ and $\mathbb{P}(X_1, X_2, X_3)$, we obtain the factorization of the embedding of $\mathbb{P}(X_1, X_2)$ (second order embedding)

$$\begin{aligned} \mathcal{C}_{X_1 X_2} &= \sum\nolimits_{h \in [k]} \left(\mu_{X_1|h} \otimes \mu_{X_2|h} \right) \mathbb{P}(h) \\ &= \sum\nolimits_{h \in [k]} \left(\mathcal{C}_{X|H} e_h \right) \otimes \left(\mathcal{C}_{X|H} e_h \right) \mathbb{P}(h) \\ &= \mathcal{C}_{X|H} \left(\sum\nolimits_{h \in [k]} e_h \otimes e_h \, \mathbb{P}(h) \right) \mathcal{C}_{X|H}^\top \\ &= \mathcal{C}_{X|H} \, \mathcal{C}_{HH} \, \mathcal{C}_{X|H}^\top, \end{aligned} \tag{2.8}$$

and that of $\mathbb{P}(X_1, X_2, X_3)$ (third order embedding)

$$\mathcal{C}_{X_1 X_2 X_3} = \mathcal{C}_{HHH} \times_1 \mathcal{C}_{X|H} \times_2 \mathcal{C}_{X|H} \times_3 \mathcal{C}_{X|H}. \tag{2.9}$$

2.4.3 Identifiability of Parameters

We note that $\mathcal{C}_{X|H} = \big(\mu_{X|h=1},\ \mu_{X|h=2},\ \ldots,\ \mu_{X|h=k}\big)$, and the kernel embeddings for $\mathcal{C}_{X_1X_2}$ and $\mathcal{C}_{X_1X_2X_3}$ can be alternatively written as

$$\mathcal{C}_{X_1X_2} = \sum\nolimits_{h\in[k]} \pi_h \cdot \mu_{X|h} \otimes \mu_{X|h}, \tag{2.10}$$

$$\mathcal{C}_{X_1X_2X_3} = \sum\nolimits_{h\in[k]} \pi_h \cdot \mu_{X|h} \otimes \mu_{X|h} \otimes \mu_{X|h}. \tag{2.11}$$

[21] showed that, under mild conditions, a finite mixture of nonparametric product distributions is identifiable. The multi-view latent variable model in (2.10) and (2.11) has the same form as a finite mixture of nonparametric product distribution, and therefore we can adapt Allman's results to the current setting.

Proposition 1 (Identifiability) *Let $\mathbb{P}(X_1, X_2, X_3)$ be a multi-view latent variable model, such that the conditional distributions $\{\mathbb{P}(X|h)\}_{h\in[k]}$ are linearly independent. Then, the set of parameters $\{\pi_h, \mu_{X|h}\}_{h\in[k]}$ are identifiable from $\mathcal{C}_{X_1X_2X_3}$, up to label swapping of the hidden variable H.*

Example 1. The probability vector of a discrete variable $X \in [n]$, and the joint probability table of two discrete variables $X_1 \in [n]$ and $X_2 \in [n]$, are both kernel embeddings. To see this, let the kernel be the Kronecker delta kernel $\kappa(x, x') = \delta(x, x')$ whose feature map $\phi(x)$ is the standard basis of e_x in $\mathbb{R}^n$. The x-th dimension of e_x is 1 and 0 otherwise. Then

$$\mu_X = \begin{pmatrix} \mathbb{P}(x=1) & \ldots & \mathbb{P}(x=n) \end{pmatrix}^\top,$$
$$\mathcal{C}_{X_1X_2} = \Big(\mathbb{P}(x_1 = s, x_2 = t) \Big)_{s,t\in[n]}.$$

We require that the conditional probability table $\{P(X|h)\}_{h\in[k]}$ to have full column rank for identifiability in this case.

Example 2. Suppose we have a k-component mixture of one dimensional spherical Gaussian distributions. The Gaussian components have identical covariance σ^2, but their mean values are distinct. Note that this model is not identifiable under the framework of [12] since the mean values are just scalars and therefore, rank deficient. However, if we embed the density functions using universal kernels such as Gaussian RBF kernel, it can be shown that the mixture model becomes identifiable. This is because we are working with the entire density function which are linearly independent from each other in this case. Thus, the non-parametric framework allows us to incorporate a wider range of latent variable models.

Finally, we remark that the identifiability result in Proposition 1 can be extended to cases where the conditional distributions do not satisfy linear independence, *i.e.*, they are overcomplete, e.g. [22, 23, 24]. However, in general, it is not tractable to learn such overcomplete models and we do not consider them here.

2.5 Kernel Algorithm

We first design a kernel algorithm to recover the parameters, $\left\{\pi_h, \mu_{X|h}\right\}_{h\in[k]}$, of the multi-view latent variable model based on $\mathcal{C}_{X_1X_2}$ and $\mathcal{C}_{X_1X_2X_3}$. This can be easily extended to the sample versions and this is discussed in Section 2.5.2. Again for simplicity of exposition, the algorithm is explained for symmetric view case. The more general version is presented in Appendix A.1.

2.5.1 Population Case

We first derive the algorithm for the population case as if we could access the true operator $\mathcal{C}_{X_1X_2}$ and $\mathcal{C}_{X_1X_2X_3}$. Its finite sample counterpart will be presented in the next section. The algorithm can be thought of as a kernel generalization of the algorithm in [25] using embedding representations.

Step 1. We perform eigen-decomposition of $\mathcal{C}_{X_1X_2}$,

$$\mathcal{C}_{X_1X_2} = \sum\nolimits_{i=1}^{\infty} \sigma_i \cdot u_i \otimes u_i$$

where the eigen-values are ordered in non-decreasing manner. According to the factorization in Eq. (2.8), $\mathcal{C}_{X_1X_2}$ has rank k. Let the leading eigenvectors corresponding to the largest k eigen-value be $\mathcal{U}_k := (u_1, u_2, \ldots, u_k)$, and the eigen-value matrix be $S_k := \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$. We define the whitening operator $\mathcal{W} := \mathcal{U}_k S_k^{-1/2}$ which satisfies

$$\mathcal{W}^\top \mathcal{C}_{X_1X_2} \mathcal{W} = (\mathcal{W}^\top \mathcal{C}_{X|H} \mathcal{C}_{HH}^{1/2})(\mathcal{C}_{HH}^{1/2} \mathcal{C}_{X|H}^\top \mathcal{W}) = I,$$

and $M := \mathcal{W}^\top \mathcal{C}_{X|H} \mathcal{C}_{HH}^{1/2}$ is an orthogonal matrix.

Step 2. We apply the whiten operator to the 3rd order kernel embedding $\mathcal{C}_{X_1X_2X_3}$

$$\mathcal{T} := \mathcal{C}_{X_1X_2X_3} \times_1 (\mathcal{W}^\top) \times_2 (\mathcal{W}^\top) \times_3 (\mathcal{W}^\top).$$

According to the factorization in Eq. (2.9), $\mathcal{T} = \mathcal{C}_{HHH}^{-1/2} \times_1 M \times_2 M \times_3 M$, which is a tensor with orthogonal factors. Essentially, each column v_i of M is an eigenvector of $\mathcal{T}$.

Step 3. We use tensor power method to find the leading k eigenvectors M for $\mathcal{T}$ [10]. The corresponding k eigenvalues $\lambda = (\lambda_1, \ldots, \lambda_k)^\top$ will then be equal to

$$(\mathbb{P}(h = 1)^{-1/2}, \ldots, \mathbb{P}(h = k)^{-1/2}).$$

The tensor power method is provided in the Appendix in Algorithm 8 for completeness.

Step 4. We recover the conditional embedding operator by undoing the whitening step

$$\mathcal{C}_{X|H} = (\mathcal{W}^\top)^\dagger M \, \mathrm{diag}(\lambda).$$

2.5.2 Finite Sample Case

Given m observation $\mathcal{D}_{X_1X_2X_3} = \{(x_1^i, x_2^i, x_3^i)\}_{i\in[m]}$ drawn *i.i.d.* from a multi-view latent variable model $\mathbb{P}(X_1, X_2, X_3)$, we now design a kernel algorithm to estimate the latent parameters from data. Although the empirical kernel embeddings can be infinite dimensional, we can carry out the decomposition using just the kernel matrices. We denote the implicit feature matrix by

$$\Phi := (\phi(x_1^1), \ldots, \phi(x_1^m), \phi(x_2^1), \ldots, \phi(x_2^m)),$$
$$\Psi := (\phi(x_2^1), \ldots, \phi(x_2^m), \phi(x_1^1), \ldots, \phi(x_1^m)),$$

and the corresponding kernel matrix by $K = \Phi^\top\Phi$ and $L = \Psi^\top\Psi$ respectively. And we denote $K_{:x} := \Phi^\top\phi(x)$ as a column vector containing the kernel between x and data points in Φ. For three vectors ξ_1, ξ_2 and ξ_3, denote the symmetric tensor obtained from their outer product

$$\otimes[\xi_1, \xi_2, \xi_3] := \xi_1 \otimes \xi_2 \otimes \xi_3 + \xi_3 \otimes \xi_1 \otimes \xi_2 + \xi_2 \otimes \xi_3 \otimes \xi_1.$$

Then the steps in the population case can be mapped one-by-one into kernel operations.

Step 1. We perform a kernel eigenvalue decomposition of the empirical 2nd order embedding

$$\widehat{\mathcal{C}}_{X_1X_2} := \frac{1}{2m}\sum\nolimits_{i=1}^{m}\left(\phi(x_1^i)\otimes\phi(x_2^i) + \phi(x_2^i)\otimes\phi(x_1^i)\right),$$

which can be expressed succinctly as $\widehat{\mathcal{C}}_{X_1X_2} = \frac{1}{2m}\Phi\Psi^\top$. Its leading k eigenvectors $\widehat{\mathcal{U}}_k = (\widehat{u}_1, \ldots, \widehat{u}_k)$ lie in the span of the column of Φ, *i.e.*, $\widehat{\mathcal{U}}_k = \Phi(\beta_1, \ldots, \beta_k)$ with $\beta \in \mathbb{R}^{2m}$. Then we can transform the eigen-value decomposition problem for an infinite dimensional matrix to a problem involving finite dimensional kernel matrices,

$$\begin{aligned}\widehat{\mathcal{C}}_{X_1X_2}\widehat{\mathcal{C}}_{X_1X_2}^\top u = \widehat{\sigma}^2\, u \;&\Rightarrow\; \frac{1}{4m^2}\Phi\Psi^\top\Psi\Phi^\top\Phi\beta = \widehat{\sigma}^2\,\Phi\beta \\ &\Rightarrow\; \frac{1}{4m^2}KLK\beta = \widehat{\sigma}^2\,K\beta.\end{aligned}$$

Let the Cholesky decomposition of K be $R^\top R$. Then by redefining $\widetilde{\beta} = R\beta$, and solving an eigenvalue problem

$$\frac{1}{4m^2} RLR^\top \widetilde{\beta} = \widehat{\sigma}^2 \widetilde{\beta}, \text{ and obtain } \beta = R^\dagger \widetilde{\beta}. \tag{2.12}$$

The resulting eigenvectors satisfy $u_i^\top u_{i'} = \beta_i^\top \Phi^\top \Phi \beta_{i'} = \beta_i^\top K \beta_{i'} = \widetilde{\beta}_i^\top \widetilde{\beta}_{i'} = \delta_{ii'}$.

Step 2. We whiten the empirical 3rd order embedding

$$\widehat{\mathcal{C}}_{X_1X_2X_3} := \frac{1}{3m} \sum\nolimits_{i=1}^{m} \otimes \left[\phi(x_1^i), \phi(x_2^i), \phi(x_3^i)\right]$$

using $\widehat{\mathcal{W}} := \widehat{\mathcal{U}}_k \widehat{S}_k^{-1/2}$, and obtain

$$\widehat{\mathcal{T}} := \frac{1}{3m} \sum\nolimits_{i=1}^{m} \otimes \left[\xi(x_1^i), \xi(x_2^i), \xi(x_3^i)\right]$$

where $\xi(x_1^i) := \widehat{S}_k^{-1/2} (\beta_1, \ldots, \beta_k)^\top K_{:x_1^i} \in \mathbb{R}^k$.

Step 3. We run tensor power method [10] on the finite dimension tensor $\widehat{\mathcal{T}}$ to obtain its leading k eigenvectors $\widehat{M} := (\widehat{v}_1, \ldots, \widehat{v}_k)$ and the corresponding eigenvalues $\widehat{\lambda} := (\widehat{\lambda}_1, \ldots, \widehat{\lambda}_k)^\top$.

Step 4. The estimates of the conditional embeddings are

$$\widehat{\mathcal{C}}_{X|H} = \Phi(\beta_1, \ldots, \beta_k) \widehat{S}_k^{1/2} \widehat{M} \operatorname{diag}(\widehat{\lambda}).$$

The overall kernel algorithm is summarized in Algorithm 1.

2.6 Sample Complexity

Let $\rho := \sup_{x \in \mathcal{X}} \kappa(x, x)$, $\| \cdot \|$ be the Hilbert-Schmidt norm, $\pi_{\min} := \min_{i \in [k]} \pi_i$ and $\sigma_k(\mathcal{C}_{X_1X_2})$ be the k-th largest singular value of $\mathcal{C}_{X_1X_2}$. In the following, we provide sample complexity bounds for the estimated conditional embedding $\mu_{X|h}$ and the corresponding

Algorithm 1 Kernel Spectral Algorithm

In: Kernel matrices K and L, and desired rank k
Out: A vector $\widehat{\pi} \in \mathbb{R}^k$ and a matrix $A \in \mathbb{R}^{2m \times k}$

1: Cholesky decomposition: $K = R^\top R$
2: Eigen-decomposition: $\frac{1}{4m^2} RLR^\top \widetilde{\beta} = \widehat{\sigma}^2 \, \widetilde{\beta}$
3: Use k leading eigenvalues: $\widehat{S}_k = \mathrm{diag}(\widehat{\sigma}_1, \ldots, \widehat{\sigma}_k)$
4: Use k leading eigenvectors $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$ to compute: $(\beta_1, \ldots, \beta_k) = R^\dagger(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$
5: Form tensor: $\widehat{\mathcal{T}} = \frac{1}{3m} \sum_{i=1}^m \otimes [\xi(x_1^i), \xi(x_2^i), \xi(x_3^i)]$ where $\xi(x_1^i) = \widehat{S}_k^{-1/2} (\beta_1, \ldots, \beta_k)^\top K_{:x_1^i}$
6: Power method: eigenvectors $\widehat{M} := (\widehat{v}_1, \ldots, \widehat{v}_k)$, and the eigenvalues $\widehat{\lambda} := (\widehat{\lambda}_1, \ldots, \widehat{\lambda}_k)^\top$ of $\widehat{\mathcal{T}}$
7: $A = (\beta_1, \ldots, \beta_k) \widehat{S}_k^{1/2} \widehat{M} \, \mathrm{diag}(\widehat{\lambda})$
8: $\widehat{\pi} = (\widehat{\lambda}_1^{-2}, \ldots, \widehat{\lambda}_k^{-2})^\top$

prior distribution π (the proof is in Appendix A.3).

Theorem 2 *Pick any $\delta \in (0, 1)$. When the number of samples m satisfies*

$$m > \frac{\theta \rho^2 \log \frac{2}{\delta}}{\sigma_k^2(\mathcal{C}_{X_1 X_2})}, \quad \theta := \max\left(\frac{C_3 k^2 \rho}{\sigma_k(\mathcal{C}_{X_1 X_2})}, \frac{C_4 k^{2/3}}{\pi_{\min}^{1/3}} \right),$$

for some constants $C_3, C_4 > 0$, and the number of iterations N and the number of random initialization vectors L (drawn uniformly on the sphere $\mathcal{S}^{k-1}$) satisfy

$$N \geq C_2 \cdot \left(\log(k) + \log\log\Big(\frac{1}{\sqrt{\pi_{\min}} \epsilon_{\mathcal{T}}} \Big) \right),$$

for constant $C_2 > 0$ and $L = \mathrm{poly}(k) \log(1/\delta)$, the robust power method in [10] yields eigen-pairs $(\widehat{\lambda}_i, \widehat{v}_i)$ such that there exists a permutation η, with probability $1 - 4\delta$, we have

$$\|\pi_j^{-1/2} \mu_{X|h=j} - (\beta_1, \ldots, \beta_k) \widehat{S}_k^{1/2} \widehat{v}_{\eta(j)}\|_{\mathcal{F}} \leq 8 \epsilon_{\mathcal{T}} \cdot \pi_j^{-1/2},$$
$$|\pi_j^{-1/2} - \widehat{\lambda}_{\eta(j)}| \leq 5 \epsilon_{\mathcal{T}}, \quad \forall j \in [k],$$

and $\|\mathcal{T} - \sum_{j=1}^{k} \hat{\lambda}_j \hat{\phi}_j^{\otimes 3}\| \leq 55\epsilon_{\mathcal{T}}$ *where* $\epsilon_{\mathcal{T}} := \|\widehat{\mathcal{T}} - \mathcal{T}\|$ *is the tensor perturbation bound*

$$\epsilon_{\mathcal{T}} \leq \frac{8\rho^{1.5}\sqrt{\log\frac{2}{\delta}}}{\sqrt{m}\,\sigma_k^{1.5}(\mathcal{C}_{X_1X_2})} + \frac{512\sqrt{2}\rho^3\left(\log\frac{2}{\delta}\right)^{1.5}}{m^{1.5}\,\sigma_k^3(\mathcal{C}_{X_1X_2})\sqrt{\pi_{\min}}}$$

Proof Sketch: Our proof is different from those in [10] which only analyze the perturbation of the tensor decomposition. Our proof further takes into account the error introduced by the approximate whitening step, and its effects to the tensor decomposition. ■

Remark 1: We note that the sample complexity is $\mathrm{poly}(k, \rho, 1/\pi_{\min}, 1/\sigma_k(\mathcal{C}_{X_1X_2}))$ of a low order, and in particular, it is $O(k^2)$, when the other parameters are fixed. For the special case of discrete measurements, where the kernel $\kappa(x, x') = \delta(x, x')$, we have $\rho = 1$. Note that the sample complexity depends in this case only on the number of components k and not on the dimensionality of the observed state space.

Remark 2: Theorem 2 also gives us an error bound for estimating the integral of a function $f \in \mathcal{F}$ with respect to a mixture component in unsupervised fashion. Under the conditions specified in the theorem, we have

$$\begin{aligned}&\left|\int_{\mathcal{X}} f(x)\, d\mathbb{P}(x|h) - \left\langle f, \widehat{\mu}_{X|h}\right\rangle_{\mathcal{F}}\right| \\ &\leq \|f\|_{\mathcal{F}} \left\|\mu_{X|h} - \widehat{\mu}_{X|h}\right\|_{\mathcal{F}} = O\left(\tfrac{1}{\sqrt{m}}\right)\end{aligned}$$

assuming $\|f\|_{\mathcal{F}}$ is bounded and $\rho/\sigma_k = O(1)$. We are not aware of any other result of the similar type in this unsupervised setting.

Remark 3: For $x \in \mathbb{R}^d$ and the normalized Gaussian RBF kernel in (2.2), the recovered conditional embedding $\widehat{\mu}_{X|h}$ can be used to estimate the conditional density, $p(x|h) \approx$

$\widehat{p}(x|h) := \left\langle \phi(x), \widehat{\mu}_{X|h} \right\rangle_{\mathcal{F}}$. In this case, the error can be decomposed into two terms

$$|p(x|h) - \widehat{p}(x|h)| \leq \underbrace{|p(x|h) - \widetilde{p}(x|h)|}_{O(s^2) \text{ bias as in (2.7)}} + \underbrace{|\widetilde{p}(x|h) - \widehat{p}(x|h)|}_{\text{estimation error}}$$

where s is kernel bandwidth and $\widetilde{p}$ is the density convolved with the kernel function. The estimation error is bounded by $|\widetilde{p}(x|h) - \widehat{p}(x|h)| \leq \|\phi(x)\|_{\mathcal{F}} \|\mu_{X|h} - \widehat{\mu}_{X|h}\|_{\mathcal{F}} = O(\rho^{1/2} \cdot m^{-1/2}) = O(s^{-d/2} m^{-1/2})$ assuming $\rho/\sigma_k = O(1)$ and using $\rho = O(s^{-d})$. Under the conditions specified in Theorem 2, we combine the analysis for the two sources of errors, and obtain the bound

$$|p(x|h) - \widehat{p}(x|h)| = O(s^2 + s^{-d/2} m^{-1/2})$$

Then we have $|p(x|h) - \widehat{p}(x|h)| = O(m^{-2/(4+d)})$ if we balance the two terms by setting $s = O(m^{-1/(4+d)})$. We are not aware of any other result of the similar type in this unsupervised setting.

2.7 Discussion

Our algorithm and theoretical results can also be generalized to the settings of latent variable models with Dirichlet priors and nonparametric independent component analysis (ICA) as in [10]. In the first setting, a Dirichlet prior is placed on the mixing weights π of the multi-view latent variables, $\mathbb{P}(\pi) = \frac{\Gamma(\theta_0)}{\prod_{i \in [k]} \Gamma(\theta_i)} \prod_{i \in [k]} \pi_i^{\theta_i - 1}$ where $\theta_0 = \sum_{i \in [k]} \theta_i$ with $\theta_i > 0$, and $\Gamma(\cdot)$ is the Gamma function. In this case, we only need to modify the second and third order kernel embedding $\mathcal{C}_{X_1 X_2}$ and $\mathcal{T}$ respectively, and then Algorithm 1 applies. In the nonparametric ICA setting, the feature map $\phi(X)$ of an observed variable X is assumed to be generated from a latent vector $H \in \mathbb{R}^k$ with independent coordinates via an operator $\mathcal{A} : \mathbb{R}^k \mapsto \mathcal{F}$, $\phi(X) := \mathcal{A}\, H + Z$, where Z is a zero mean random vector independent of H. In this case, we need to start with a modified 4-th order kernel embedding,

and then reduce to a multi-view problem and estimate $\mathcal{A}$ via Algorithm 1.

2.8 Experiments

Methods. We compared our kernel spectral algorithm with four alternatives

1. The EM algorithm for mixture of Gaussians. The EM algorithm is not guaranteed to find the global solution in each trial. Thus we randomly initialize it 10 times.
2. The EM-like algorithm for mixture of nonparametric densities [15]. We initialize the algorithm with k-means as [15].
3. The spectral algorithm for mixture of spherical Gaussians [12]. Their assumption is restrictive: the centers of the Gaussian need to span a k-dimension subspace, thus it is not applicable for rank deficiency case where $k \geq l$.
4. A discretization based spectral algorithm [26]. This algorithm approximates the joint distribution of the observed variables with histogram and then applies the spectral algorithm to recover the discretized conditional density.

Both our method and the [15] have a hyper-parameter, kernel bandwidth, which we selected for each view separately using cross-validation.

2.8.1 Synthetic Data

We generated three-dimensional synthetic data from various mixture models. The variables corresponding to the dimensions are independent given the latent component indicator. More specifically, we explored two settings (1) Gaussian conditional densities with different variances; (2) Mixture of Gaussian and shifted Gamma conditional densities. The shifted Gamma distribution has density

$$p(x|h) = \frac{(x-\mu)^{(d-1)}e^{-x/\theta}}{\theta^d \Gamma(d)},\ x \geq \mu$$

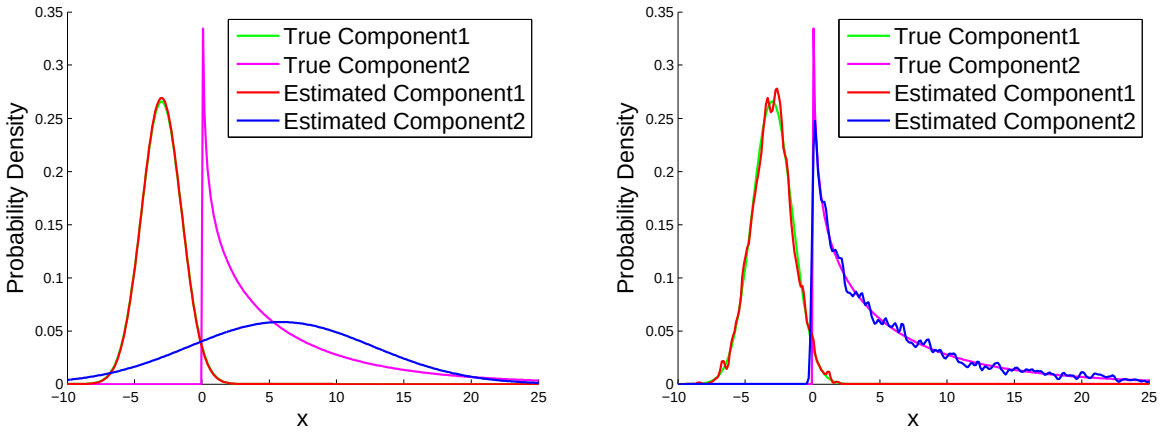
where we chose the shape parameter $d \leq 1$ such that density is very skewed. Furthermore, we chose the mean and variance parameters of the Gaussian/Gamma density such that component pair-wise overlap is relatively small according to the Fisher ratio $\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}$.

We also varied the number of samples m for the observed variables X_1, X_2 and X_3 from 50 to $10,000$, and experimented with $k = 2, 3, 4$ or 8 mixture components. The mixture proportion for the h-th component is set to be $\pi_h = \frac{2h}{\kappa(k+1)}, \ \forall h \in [k]$ (unbalanced). It is worth noting that as k becomes larger, it is more difficult to recover parameters. This is because only a small number of data will be generated for the first several clusters. For every n, k in each setting, we randomly generated 10 sets of samples and reported the average results. We note that the values for the latent variables are not given to the algorithms, and hence this is an unsupervised setting to recover the conditional density $p(x|h)$ and the ratio $p(h)$.

Error measure. We measured the performance of algorithms by the following weighted ℓ_2 norm difference $MSE := \sum_{h=1}^{k} \pi_h \sqrt{\sum_{j=1}^{m'} (p(x^j|h) - \widehat{p}(x^j|h))^2}$, where $\{x^j\}_{j\in[m]}$ is a set of uniformly-spaced test points.

Results. We first illustrated the actual recovered conditional densities of our method and EM-GMM in Figure 2.2 as a concrete example. The kernel spectral algorithm recovers nicely both the Gaussian and Gamma components, while the EM-GMM fails to fit the Gamma component.

More quantitative results are plotted in Figure 2.3. It is clear that the kernel spectral method converges rapidly with the data increment in all experiment settings. In the mixture of Gaussians setting, the EM algorithm is best since the model is correctly specified. The spectral algorithm for spherical Gaussians does not perform well since the assumption of the method is too restricted. The performance of our kernel method converges to that of the EM algorithm. In the mixture of Gaussian and Gamma setting, our kernel spectral algorithm achieves superior results compared to other algorithms. These results demonstrate that our algorithm is able to automatically adapt to the shape of the density.



(a) EM Gaussians Mixture

(b) Kernel Spectral

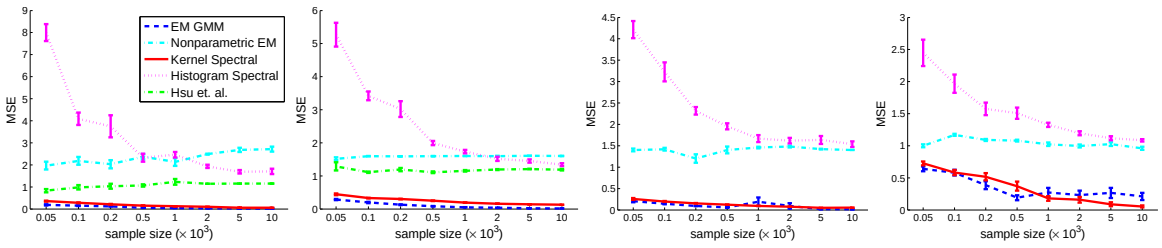
Figure 2.2: Kernel spectral algorithm is able to adapt to the shape of the mixture components, while EM algorithm for mixture of Gaussians misfits the Gamma distribution.

It is worth noting that both the discretized spectral algorithm and nonparametric EM-like algorithm did not perform as well. In the discretized spectral method, the joint distribution is estimated by histogram. It is well-known that the histogram estimation suffers from poor performance even for 3 dimensional data. In the nonparametric EM-like algorithm, besides the issue of local minima, its performance also highly depends on the initialization. And the flexibility of nonparametric densities without regularization makes the issue of overfitting quite severe, often leading to a single component in the algorithm.

We also note that the our method outperforms the EM-GMM more as the number of components increases. This is the key advantage of our method in that it has favorable performance in higher dimensions, which agrees with the theoretical result in Theorem 2 that the sample complexity depends only quadratically in the number of components, when other parameters are held fixed.

2.8.2 Flow Cytometry Data

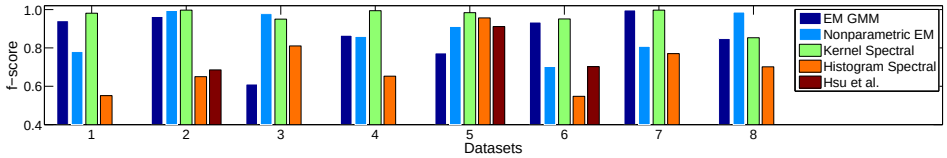
Flow cytometry (FCM) data are multivariate measurements from flow cytometers that record light scatter and fluorescence emission properties of hundreds of thousands of individual cells. They are important to the studying of the cell structures of normal and ab-



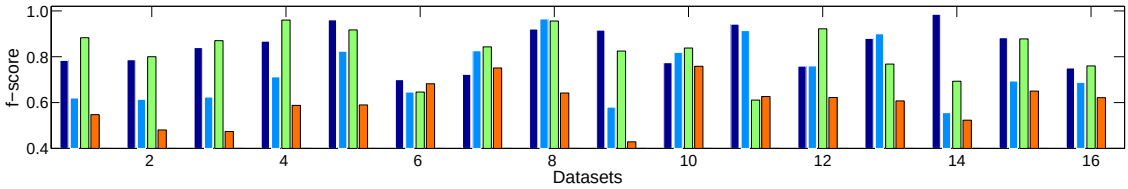
(a) Same $k = 2$ (b) Same $k = 3$ (c) Same $k = 4$ (d) Same $k = 8$

(e) Diff. $k = 2$ (f) Diff. $k = 3$ (g) Diff. $k = 4$ (h) Diff. $k = 8$

Figure 2.3: (a)-(d) Mixture of Gaussian distributions with $k = 2, 3, 4, 8$ components. (e)-(h) Mixture of Gaussian/Gamma distribution with $k = 2, 3, 4, 8$. For the former case, the performances of kernel spectral algorithm converge to those of EM algorithm for mixture of Gaussian model. For the latter case, the performances of kernel spectral algorithm are consistently much better than EM algorithm for mixture of Gaussian model. Spherical Gaussian spectral algorithm does not work for $k = 4, 8$ since $k > l(= 3)$ causes rank deficiency.



(a) number of clusters $k = 2$



(b) number of clusters $k = 3$

Figure 2.4: Clustering results on the datasets from the DLBCL flow cytometry data. The results for spherical Gaussian spectral algorithm (Hsu et al.) are not plotted for datasets on which it has rank deficiency problem. The datasets are ordered by increasing sample size.

normal cells and the diagnosis of human diseases. [27] introduced the FlowCAP-challenge whose main task is grouping the flow cytometry data automatically. Clustering on the FCM data is a difficult task because the distribution of the data is non-Gaussian and heav-

ily skewed. We use the DLBCL Lymphoma dataset collection from [27] to compare our kernel algorithm with the four alternatives. This collection contains 24 datasets with two or three clusters, and each dataset consists of tens of thousands of cell measurements in 5 dimensions. Each dataset is a separate clustering task, so we fit a multi-view model to each dataset separately and use the maximum-a-posteriori assignment to obtain the cluster labels. All the cell measurements have been manually labeled, therefore we can evaluate the clustering performance using f-score [27].

We split the 5 dimensions into three views: dimension 1 and 2 as the first view, 3 and 4 the second, and 5 the third view based on correlation between views, since we would like the views to satisfy the conditional independence assumptions to ensure good performance for the kernel spectral method. For each dataset, we select the best kernel bandwidth by 5-fold cross validation using log-likelihood. Figure 2.4 presents the results sorted by the number of clusters. Since the data are collapsed in most cases, the centers cannot span a subspace with enough rank. Thus, the method in [12] is not applicable. However, our method (kernel spectral) outperforms EM-GMM as well as the other algorithms in a majority of datasets. There are also datasets where kernel spectral algorithm has a large gap in performance compared to GMM. These are the datasets where the multi-view assumptions are heavily violated. For example, in some datasets, the correlation coefficient between dimensions 3 and 5 is as high as 0.927 given a particular cluster label, suggesting strong correlation between the two views. Obtaining improved and robust performance in these datasets will be a subject of our future study where we plan to develop even more robust kernel spectral algorithms.

CHAPTER 3

SPECTRAL LATENT FEATURES FOR DNA MOTIF PREDICTION

In this chapter, we demonstrate one application of spectral methods for latent variable estimation. We model the DNA sequence as a HMM, and then use spectral methods to learn the representation. In this particular task, we use the DNA sequence to predict the Polyadenylation motif, which is the addition of a poly(A) tail to an RNA molecule. Identifying DNA sequence motifs that signal the addition of poly(A) tails is essential to improved genome annotation and better understanding of the regulatory mechanisms and stability of mRNA.

Existing poly(A) motif predictors demonstrate that information extracted from the surrounding nucleotide sequences of candidate poly(A) motifs can differentiate true motifs from the false ones to a great extent. A variety of sophisticated features has been explored, including sequential, structural, statistical, thermodynamic and evolutionary properties. However, most of these methods involve extensive manual feature engineering, which can be time-consuming and can require in-depth domain knowledge.

We propose a novel method for poly(A) motif prediction by marrying generative learning (hidden Markov models) and discriminative learning (support vector machines). Here, we employed hidden Markov models for fitting the DNA sequence dynamics, and developed an efficient spectral algorithm for extracting latent variable information from these models. These spectral latent features were then fed into support vector machines to fine tune the classification performance.

We evaluated our proposed method on a comprehensive human poly(A) dataset that consists of 14,740 samples from 12 of the most abundant variants of human poly(A) motifs. Compared with one of the previous state-of-the-art methods in the literature (the random forest model with expert-crafted features), our method reduces the average error rate, false negative rate and false positive rate by 26%, 15% and 35%, respectively. Meanwhile, our

method made about 30% fewer error predictions relative to the other string kernels. Furthermore, our method can be used to visualize the importance of oligomers and positions in predicting poly(A) motifs, from which we can observe a number of characteristics in the surrounding regions of true and false motifs that have not been reported before.

3.1 Introduction

The poly(A) signal prediction problem has been studied for decades [28]. A number of studies have demonstrated that information from relatively short upstream and downstream sequences of the candidate poly(A) motifs can specify the true poly(A) motifs to a great extent [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. Statistical properties of the surrounding sequences were explored in different species, such as yeast [40], fly [41], Arabidopsis and rice [37], and human [30, 41, 38]. Although significant progress has been made to the accuracy of poly(A) motif predictors, especially in human DNA sequences, such methods are all based on utilizing sophisticated features that require additional efforts to extract and are highly dependent on domain knowledge.

Automatic feature extraction techniques have been explored in many other sequence classification problems. By far, the most successful methods are hidden Markov models, e.g., in gene finding [42, 43] and string kernels with support vector machines, e.g., in protein classification [44], transcription start site recognition [45] and splice site prediction [46]. The former methods are generative models that capture the uncertainty in data using probabilistic languages, while the latter are discriminative methods that optimize specifically for the classification results. The advantages of each class of methods have rarely been combined systematically to yield even better feature extractors.

In this chapter, we propose a novel method for poly(A) motif prediction by marrying generative learning (hidden Markov models) and discriminative learning (support vector machines). Generative learning provides us with a rich palette for handling the uncertainty and diversity of sequence information, while discriminative learning allows us to directly

optimize performance for the classification task. Here, *diversity* means that for the same position in the surrounding sequence of a candidate poly(A) motif, there may be multiple subsequences indicating the class label (a true motif or a false one), and *uncertainty* means that each of these subsequences does not give deterministic information about the class label. In particular, we employ hidden Markov models (HMMs) as a probabilistic generative model for DNA sequences, and develop an efficient spectral algorithm for extracting latent variable information from these models. The HMMs model not only the diversity and uncertainty of sequence information, but also long-range dependencies between subsequences at different positions, which cannot be simultaneously captured by string kernels as is discussed in Section 3.2.2. The spectral latent features are then fed into support vector machines to fine tune the classification performance.

3.2 Related Works

We will first review the two classes of methods, expert-crafted features and string kernels, for addressing the poly(A) motif classification problem.

3.2.1 Features based on domain knowledge

Bioinformatics experts can craft highly informative features based on prior knowledge of the biology and physics of DNA sequences. The drawback of using these features is that they require extensive expert domain knowledge for their design as well as additional efforts to extract them.

Salamov and Solovyev developed POLYAH [29], a tool that used a linear discriminant function-based classifier to extract features from 100 nt upstream and 200 nt downstream of a candidate poly(A) motif. Later on, polyadq was developed based on quadratic discriminant functions by encoding features from 100 nt downstream only [30]. Several support vector machine-based predictors were then proposed and they performed well on recognizing true poly(A) motifs. Such methods include DNAFSMiner, which was based on both

100 nt upstream and downstream [33], Polya_svm, which encoded *cis*-regulatory element features [34], and polyApred, which extracted features from 100 nt upstream and downstream of the candidate poly(A) motifs [35]. In 2010, Akhtar et al. proposed POLYAR [36], a linear discriminant analysis-based method that used features from 300 nt upstream and downstream of the candidate poly(A) motifs.

Recently, Kalkatawi et al. developed an artificial neural network (ANN)-based method and a random forest (RF)-based method to predict poly(A) motifs in human DNA sequences [39]. Their methods were based on a variety of expert-crafted features, such as thermodynamic and structural features of dinucleotides, electron-ion interaction potentials, and position weight matrices of upstream and downstream regions relative to the candidate poly(A) motifs. In total, they extracted 274 features. They compiled a large-scale benchmark set that contained 14,740 sequences for the 12 main variants of human poly(A) motifs. The ANN and RF models significantly outperformed all previously reported studies.

3.2.2 String kernels

String kernels are positive definite functions to compute similarity between two sequences, which is then used in support vector machines to learn classifiers. These kernels essentially map sequences into high-dimensional feature spaces corresponding to subsequences and then compute inner products between two feature vectors. The drawback of string kernels is that they simply count raw sequence matches and do not explicitly take into account the uncertainty and diversity of sequence information. Many string kernels have been designed over the years, but so far few have effectively made use of generative models to deal with data uncertainty.

More specifically, given an alphabet Σ, here the DNA nucleotides $\Sigma = \{\mathtt{A}, \mathtt{G}, \mathtt{C}, \mathtt{T}\}$, let $\mathbf{x} \in \Sigma^k$ be a sequence of length k (or k-mer). The k-spectrum kernel $\kappa(\mathbf{x}, \mathbf{y})$ counts pairs of identical k-mers between two sequences $\mathbf{x}$ and $\mathbf{y}$ (of length L and L' respectively) independently of their position [47]: $\kappa(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{L-k+1} \sum_{t'=1}^{L'-k+1} \mathbb{I}\left\{x_{t:t+k-1} = y_{t':t'+k-1}\right\}$,

where $x_{t:t+k-1} := x_t x_{t+1} \ldots x_{t+k-1}$ denotes a subsequence of $\mathbf{x}$ that starts at position i and has length k, and $\mathbb{I}\{\cdot\}$ returns 1 if the two k-mers are the same and otherwise 0. This kernel effectively maps each sequence into a feature space where each dimension counts the number of occurrences of a particular k-mer and uses the inner product as the similarity between two sequences. To take uncertainty and diversity in k-mer features into account, one can also heuristically include counts on the mismatch of k-mers [47].

In contrast to the k-spectrum kernel, the weighted degree kernel explicitly takes into account the absolute positions of the k-mers in the sequence [48]:

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{k} \beta_l \sum_{t=1}^{L-l+1} \mathbb{I}\left\{x_{t:t+l-1} = y_{t:t+l-1}\right\}.$$

Analogously, one can also heuristically incorporate the counts for mismatches in the weighted degree kernel to account for a certain degree of sequence uncertainty and diversity.

However, heuristic ways of handling mismatches may result in underutilization of the sequence information. A principle way to deal with such uncertainty is to model mismatches as random variables. Along this direction, the probability product kernel [49] has been proposed for sequence analysis. This kernel also compares sequences of equal length and assumes that the absolute position in the sequence carries discriminative information. The key idea is to fit a probabilistic generative model (e.g., hidden Markov models) to each sequence separately, and then use an inner product between these generative models to define the kernel. As a result, the probability product kernel allows us to combine discriminative learning of support vector machines with generative modeling of data.

The seminal work on probability product kernels has several limitations. First, a different hidden Markov model is fitted to each sequence, which can lead to very poorly estimated model parameters and bury the useful signals. Second, training hidden Markov models with a traditional expectation maximization (EM) algorithm [4] can be time-consuming. Third, while the hidden variables model the "clean" signals, they are summed out and not

directly used for sequence comparison. Fourth, the label information for the training sequences is not used for defining the kernel. Due to these limitations, the probability product kernel does not perform as well as the k-spectrum kernel or the weighted degree kernel as we show in later experiments. In the following, we first describe our method, which also combines generative and discriminative learning, but overcomes the above four limitations.

3.3 Methods

Our method fits only two hidden Markov models (HMMs) for the entire training set, one for sequences in the positive class and another one for those in the negative class. Then, we use the posterior distributions of the hidden states from each sequence as our features. When learning the parameters of the HMMs, we employ an efficient spectral algorithm recently proposed in machine learning [5]. The novel combination of the extracted spectral latent features and support vector machines leads to state-of-the-art results in poly(A) motif prediction.

3.3.1 Sequence latent features

We use HMMs to take into account the uncertainty and diversity of the sequence information and hypothesize that there is a "clean" poly(A) signal hidden in the observed sequences. The fact that each hidden variable is related to the previously observed positions allows us to accommodate long-range dependencies. We use capital letters to denote random variables and lower case letters for their instantiations.

We combine multiple adjacent nucleotides in a DNA sequence into a "mega-observation". For example, we treat the k-mer *AAT* as a single observation. Thus, each mega-observation has $n = 4^k$ possible states. Essentially, we transform a DNA sequence into a sequence of mega-observations using an overlapping sliding window of size k, and then associate each mega-observation with a variable X_t. For example, a DNA sequence of length L' is transformed to a sequence of $L = L' - k + 1$ mega-observations, with each mega-observation

being a k-mer. We estimate the HMM for these transformed sequences.

Specifically, an HMM contains a Markov chain of hidden variables $Q_{1:L} := Q_1 \dots Q_L$ that generates the observed sequence of variables $X_{1:L} = X_1 \dots X_L$. Let m and n denote the number of states for the hidden and observed variables, respectively. We can fully specify an HMM by an $m \times m$ transition probability matrix of the hidden variables, with the (i, j)-th entry $T_{ij} = \Pr(Q_{t+1} = i | Q_t = j)$, an $n \times m$ emission probability matrix $O_{ij} = \Pr(X_t = i | Q_t = j)$, and an m dimension prior distribution vector over the hidden states $\pi_i = \Pr(Q_1 = i)$. With these model parameters, we can compute the joint distribution of the hidden and observed variables as $\Pr(X_{1:L}, Q_{1:L}) = \Pr(Q_1) \prod_{t=2}^{L} \Pr(Q_{t+1} | Q_t) \prod_{t=1}^{L} \Pr(X_t | Q_t)$ and the distribution of the observed variables as $\Pr(X_{1:L}) = \sum_{q_{1:L}} \Pr(X_{1:L}, q_{1:L})$.

We define the latent feature at position t of the sequence as the posterior distribution of the hidden variable Q_t, given the sequence up to position t:

$$[\mathbf{f}_t]_i = \Pr(Q_t = i | x_{1:t}). \tag{3.1}$$

3.3.2 Efficient spectral algorithm for latent features

Traditional HMM learning algorithms try to recover the parameters π, $\mathbf{T}$ and $\mathbf{O}$. The resulting maximum likelihood estimation problem is not convex and algorithms can only find a local optimum. These parameters π, $\mathbf{T}$ and $\mathbf{O}$ characterize the relations between hidden and observed variables that cannot be directly observed during training and are usually not uniquely identifiable. However, we may *not need* to recover them exactly in order to extract latent features. Instead, it is sufficient to recover them up to some invertible transformation if we subsequently learn a linear classifier such as a support vector machine. More specifically, suppose matrix $\mathbf{A}$ of size $m \times m$ is invertible. Define the transformed

HMM parameters

$$\mathbf{h}_0 := \mathbf{A}\pi, \quad \mathbf{h}_\infty := \left(\mathbf{A}^{-1}\right)^\top \mathbf{e}, \quad \mathbf{H}_x := \mathbf{A}\mathbf{S}_x\mathbf{A}^{-1}. \tag{3.2}$$

Then we can compute the transformed latent feature as

$$\mathbf{h}_t = \frac{\mathbf{H}_{x_t}\mathbf{h}_{t-1}}{\mathbf{h}_\infty^\top \mathbf{H}_{x_t}\mathbf{h}_{t-1}} = \mathbf{A}\mathbf{f}_t, \tag{3.3}$$

since the invertible matrix $\mathbf{A}$ cancels with $\mathbf{A}^{-1}$ during matrix multiplication. Then, the latent features of the final sequence that we use in our experiments are $\mathbf{h} := (\mathbf{h}^{+\top}, \mathbf{h}^{-\top})^\top$ by concatenating the transformed features from positive and negative HMMs. It is easy to see that the transformed feature will achieve the same performance as the original one with a linear classifier since the transformation matrix can be incorporated into the classifier weight vector. That is, if we learn a binary classifier $\text{sign}(\mathbf{w}^\top \mathbf{f} + b)$, then $\text{sign}(\tilde{\mathbf{w}}^\top \mathbf{h} + b)$ with $\tilde{\mathbf{w}} = (\mathbf{A}^{-1})^\top \mathbf{w}$ will achieve the same classification accuracy.

A natural question is how to choose an $\mathbf{A}$ such that the transformed parameters $\{\mathbf{h}_0, \mathbf{h}_\infty, \mathbf{H}_x\}$ can be easier to estimate without using an EM algorithm. To solve this problem, we employ a construction of $\mathbf{A}$ by [5], which allows these transformed parameters to be estimated from just tri-gram information of the observed sequences. Formally, let (X_1, X_2, X_3) be a triple of adjacent variables in the HMM and define the marginal probabilities of observation singletons, pairs, and triples as:

$$[\mathbf{c}_1]_i = \Pr(X_1 = i), \tag{3.4}$$

$$[\mathbf{C}_{2,1}]_{ij} = \Pr(X_2 = i, X_1 = j), \tag{3.5}$$

$$[\mathbf{C}_{3,x,1}]_{ij} = \Pr(X_3 = i, X_2 = x, X_1 = j), \quad 1 \le x \le n. \tag{3.6}$$

$\mathbf{c}_1$ is an n dimensional vector, $\mathbf{C}_{2,1}$ is a $n \times n$ matrix and $\mathbf{C}_{3,x,1}$ is a series of $n \times n$ matrices indexed by x. [5] showed that setting $\mathbf{A} = \mathbf{U}^\top \mathbf{O}$ directly links the above three quantities

Algorithm 2 Spectral learning of transformed HMM parameters

Require: m - the number of hidden states, k - the number of nt to combine
Ensure: transformed HMM parameters $\{\mathbf{h}_0, \mathbf{h}_\infty, \mathbf{H}_x\}$

1: Transform all training sequences by combining k consecutive nt into a "mega-observation".
2: Use all triples (x_1, x_2, x_3) from the transformed sequences to estimate $\mathbf{c}_1$, $\mathbf{C}_{2,1}$, and $\mathbf{C}_{3,x,1}$ according to (3.9), (3.10) and (3.11).
3: Compute the SVD of $\mathbf{C}_{2,1}$, and let $\mathbf{U}$ be the matrix of left singular vectors corresponding to the m largest singular values.
4: Compute transformed model parameters using (3.7) and (3.8).

to the transformed parameters in (3.2), where $\mathbf{U}$ is the leading m principal left singular vectors of $\mathbf{C}_{2,1}$. Specifically, the transformed parameters can be directly recovered from single sequence statics as

$$\mathbf{h}_0 = \mathbf{U}^\top \mathbf{c}_1, \qquad \mathbf{h}_\infty = (\mathbf{C}_{2,1}^\top \mathbf{U})^\dagger \mathbf{c}_1, \tag{3.7}$$

$$\mathbf{H}_x = \mathbf{U}^\top \mathbf{C}_{3,x,1} (\mathbf{U}^\top \mathbf{C}_{2,1})^\dagger, \tag{3.8}$$

where $(\cdot)^\dagger$ computes the pseudo-inverse of a matrix.

The learning algorithm for HMMs is summarized in Algorithm 2. It first estimates the quantities in (3.4), (3.5) and (3.6) using training data

$$[\mathbf{c}_1]_i \approx \tfrac{1}{DL} \sum_{d=1}^{D} \sum_{t=1}^{L} \sum_{i \in \Sigma^k} \mathbb{I}\left\{x_t^d = i\right\}, \tag{3.9}$$

$$[\mathbf{C}_{2,1}]_{ij} \approx \tfrac{1}{D(L-1)} \sum_{d=1}^{D} \sum_{t=1}^{L-1} \sum_{i,j \in \Sigma^k} \mathbb{I}\left\{x_{t:t+1}^d = (i,j)\right\}, \tag{3.10}$$

$$[\mathbf{C}_{3,x,1}]_{ij} \approx \tfrac{1}{D(L-2)} \sum_{d=1}^{D} \sum_{t=1}^{L-2} \sum_{i,j,x \in \Sigma^k} \mathbb{I}\left\{x_{t:t+2}^d = (i,x,j)\right\}, \tag{3.11}$$

where D is the number of training sequences. These estimates are subsequently used to compute the transformed HMM model parameters according to (3.7) and (3.8). The algorithm has two parameters m and k that can be tuned by cross-validation. The major computation is an SVD of $\mathbf{C}_{2,1}$ and hence the name "spectral algorithm". We note that

Algorithm 3 Spectral latent feature extraction algorithm

Require: $(x_1, \ldots, x_t)$ - a test sequence, k - the number of nt to combine, $\{\mathbf{h}_0^+, \mathbf{h}_\infty^+, \mathbf{H}_x^+\}$ and $\{\mathbf{h}_0^-, \mathbf{h}_\infty^-, \mathbf{H}_x^-\}$ - learned HMM models for positive and negative classes.
Ensure: spectral latent features $\mathbf{h}$

1: Transform the input sequence by combing k adjacent nt into a "mega-observation".
2: For $t = 1 \ldots L$, compute $\mathbf{h}_t^+ = \mathbf{H}_{x_t}^+ \mathbf{h}_{t-1}^+ / \left(\mathbf{h}_\infty^{+\top} \mathbf{H}_{x_t}^+ \mathbf{h}_{t-1}^+\right)$.
3: For $t = 1 \ldots L$, compute $\mathbf{h}_t^- = \mathbf{H}_{x_t}^- \mathbf{h}_{t-1}^- / \left(\mathbf{h}_\infty^{-\top} \mathbf{H}_{x_t}^- \mathbf{h}_{t-1}^-\right)$.
4: Concatenate features $\mathbf{h} = (\mathbf{h}_1^{+\top}, \ldots, \mathbf{h}_L^{+\top}, \mathbf{h}_1^{-\top}, \ldots, \mathbf{h}_L^{-\top})^\top$.

we learn two HMMs, one for the positive class and the other for the negative class. Then, we use both HMMs to extract features for each test sequence and concatenate the features, which is summarized in Algorithm 3.

Fast implementation. The runtime of algorithm 2 is dominated by the SVD computation of an $n \times n$ matrix $\mathbf{C}_{2,1}$, and the memory requirement is dominated by storing the tri-gram statistics $\mathbf{C}_{3,x,1}$ for each x. One technical challenge is that n, the number of possible values of "mega-observation", can grow as $n = 4^k$, exponential in k. It seems, at first sight, that we may need to decompose a huge matrix $\mathbf{C}_{2,1}$, and the memory requirement for $\mathbf{C}_{3,x,1}$ is prohibitively large. However, most entries in these matrices are zero because some k-mers do not exist in the training sequences. Moreover, the total number of non-zero entries is at most the number of "mega-observations" in the training set. Taking advantage of this property, we can do sparse matrix SVD and store all the tri-gram statistics in a sparse matrix, thus facilitating efficient computation and manipulation. The computational complexity thus grows linearly with the number of "mega-observations" in the worst case.

3.3.3 Visualizing the importance of k-mers and positions

Besides accurately classifying the sequences, we are also interested in k-mers and positions that are most informative for motif classification. [50] proposed positional oligomer importance matrices (POIMs) for weighted degree kernels to analyze the importance of substrings in different locations of the sequence. Here, we also develop a technique for visualizing the importance of the k-mer at each position t for the classification problem

based on our spectral latent features.

Intuitively, we want to use the contribution of the k-mer at position t to the support vector classifier as its importance score. In particular, we make use of the margin of a training sequence in the support vector machine. For example, if most positive sequences with large positive margins all contain k-mer *AAGC* at position t, then this k-mer is important for correct classifications. More formally, let the support vector classifier learned from our spectral latent features be $\text{sign}(\mathbf{w}^\top \mathbf{h} + b)$ and let the margin corresponding to a sequence be $s(\mathbf{x}) = \mathbf{w}^\top \mathbf{h}(\mathbf{x}) + b$, where we use $\mathbf{h}(\mathbf{x})$ to indicate that the features are extracted from sequence $\mathbf{x}$. Then, we define the importance of k-mer $y_{1:k}$ at position t as

$$\alpha(y_{1:k}@t) := \sum\nolimits_{\mathbf{x}\in\Sigma^L} s(\mathbf{x}) \Pr(\mathbf{x}|x_{t:t+k-1} = y_{1:k}). \tag{3.12}$$

That is, given the sequences that have $y_{1:k}$ at position t, we compute the importance as the weighted sum of the margin of these sequences. In practice, we only have a finite number of training sequences, and we will use the finite sample average to estimate the importance score. That is, $\alpha(y_{1:k}@t) \approx \frac{1}{|\mathcal{T}(y_{1:k}@t)|} \sum_{\mathbf{x}\in\mathcal{T}(y_{1:k}@t)} s(\mathbf{x})$, where $\mathcal{T}(y_{1:k}@t)$ denotes the set of training sequences with k-mer $y_{1:k}$ occurring at position t.

Once we have computed the importance score for every k-mer at every position t, we can visualize it in a few different ways. One way is to visualize scores as a heatmap of k-mer versus sequence position. For longer k-mers, there are 4^k possible values that cannot be easily visualized. Instead, we sum the absolute values of all k-mer importance scores at each position, as is done in [50], and visualize the importance score as a function of the position t. That is, $\alpha(@t) := \sum_{y_{1:k}\in\Sigma^k} \alpha(y_{1:k}@t)$.

3.4 Results

3.4.1 Datasets

The proposed method was tested on the benchmark set proposed in [39]. The dataset contains 14,740 sequences (7,370 with true poly(A) motifs - positive samples and 7,370 with false poly(A) motifs - negative samples) for the 12 main variants of human poly(A) motifs (see Table 3.1 for these variants and their respective sizes). For each variant, the number of positive sequences is equal to the number of negative sequences. Furthermore, for each variant, the positive motifs with the surrounding sequences were extracted from human mRNA and mapped back to the human genome, whereas the same number of negative motifs were randomly selected from human chromosome 21. Each sample is a candidate 6 nt poly(A) motif surrounded by 100 nt upstream and downstream. This represents a comprehensive benchmark set for poly(A) motifs in human DNA sequences. The goal is to predict which candidate motifs are true poly(A) motifs.

3.4.2 Experimental settings

The proposed method was tested on each of the 12 datasets using five-fold cross-validation. Each dataset was randomly partitioned into five subsets, four of which were used for training and validation, and the remaining one was used for testing in each fold.

We then searched for $m \in \{2, 4, \ldots, 40\}$ and $k \in \{3, 4, 5, 6, 7\}$ using cross-validations (Figure S1). For each parameter combination and each dataset, two HMMs were learned for the positive samples and the negative ones in the training data. For each HMM, the parameters, $\{\mathbf{h}_0, \mathbf{h}_\infty, \mathbf{H}_x\}$, were learned by Algorithm 2. For each training sequence of 206 nt, the spectral latent feature vectors, $(\mathbf{h}_1^\top, \ldots, \mathbf{h}_{206-k+1}^\top)^\top$, were then calculated for the positive and negative HMMs and concatenated (Algorithm 3). A linear SVM was then trained using these spectral latent features.

Given a testing sequence of 206 nt, the spectral latent feature vectors were extracted

by Algorithm 3 using the learned HMMs for the same poly(A) motif. The concatenated feature vector was then given to the corresponding SVM model to predict whether it was a true poly(A) motif or a false one. The grid search for different parameters with respect to the training and testing errors indicated that the parameter ranges that had highest accuracy on the training set were $k = 4, 5, 6$ and $m \geq 20$ (Figure S1).

3.4.3 Comparison to other string kernels

We first compare the classification performance of the proposed method (HMM) to the previous state-of-the-art string kernels, namely, the probability product kernel (PPK), k-spectrum kernel (SPE), and weighted degree kernel (WD). The best k for these alternative kernels, except PPK, were also searched using the cross-validation between 3 to 7, and we report here the best results. In Table 3.1, all reported errors are the average over the five-fold cross-validation. It can be seen that the WD kernel compares favorably with the SPE kernel, with slightly higher false negative rate and slightly lower false positive rate. Our method, which simultaneously takes into account location information, sequence uncertainty and training labels, performs consistently and significantly better than PPK, SPE and WD. The PPK kernel has the worst results. As discussed in Section 3.2, PPK can suffer from severe overfitting by fitting each sequence to a separate HMM and it discards important discriminative information by not using the training labels.

Next we compare the runtime of different methods using the largest two variants, *AATAAA* and *ATTAAA*. Our method is significantly faster than alternatives at training time, while being comparable in speed at test time (Table 3.2). In this experiment, the training time is equal to the time for kernel (or feature) computation for training data plus that for learning SVM models; and the test time is equal to the time for kernel (or feature) computation for test data plus that for classification. At training time, PPK, SPE and WD need to compute a square kernel matrix of size $D \times D$, and the associated SVM models need to be trained in the dual form. In contrast, our method computes a feature matrix of size

Table 3.1: Comparison of the error rates of our method (HMM) with PPK, SPE and WD. "*Average*" denotes the *weighted* average of the corresponding column. "Size" denotes the number of samples for the corresponding motif variant. "Error rate" is the proportion of false results in the dataset, which equals one minus accuracy. "False negative rate" is the proportion of true poly(A) motifs that are predicted to be false, which equals one minus sensitivity. "False positive rate" is the proportion of false poly(A) motifs that are predicted to be true, which equals one minus specificity. "Rel" denotes the relative improvement of HMM with respect to SPE. The lowest error rate for each motif variant is indicated in bold. PPK could not finish running within 48 hours on *AATAAA*.

Variants	Size	Error Rate (%)				
		PPK	SPE	WD	HMM	Rel
AATAAA	5190	-	23.08	23.72	**18.59**	19.45
ATTAAA	2400	27.13	20.17	18.29	**16.21**	19.63
AAGAAA	1250	31.28	14.72	16.72	**9.36**	36.41
AAAAAG	1230	15.04	13.25	7.80	**5.45**	58.90
AATACA	880	31.48	18.98	23.18	**15.34**	19.16
TATAAA	780	29.87	16.28	18.46	**11.15**	31.50
ACTAAA	690	40.72	24.35	30.29	**16.96**	30.36
AGTAAA	670	31.19	20.90	23.88	**14.33**	31.43
GATAAA	460	25.43	17.39	14.13	**9.57**	45.00
AATATA	410	29.51	15.85	18.78	**9.27**	41.54
CATAAA	410	32.68	18.78	22.20	**12.68**	32.47
AATAGA	370	24.05	8.11	14.86	**5.14**	36.67
Average	-	-	19.56	20.22	**14.42**	28.09

Variants	Size	False Negative Rate (%)					False Positive Rate (%)				
		PPK	SPE	WD	HMM	Rel	PPK	SPE	WD	HMM	Rel
AATAAA	5190	-	21.93	23.70	**18.54**	15.47	-	24.24	23.74	**18.65**	23.05
ATTAAA	2400	32.50	22.83	21.50	**18.17**	20.44	21.75	17.50	15.08	**14.25**	18.57
AAGAAA	1250	37.12	14.08	19.68	**11.36**	19.32	25.44	15.36	13.76	**7.36**	52.08
AAAAAG	1230	25.20	8.94	8.46	**6.02**	32.73	4.88	17.56	7.15	**4.88**	72.22
AATACA	880	35.91	19.55	30.68	**19.09**	2.33	27.05	18.41	15.68	**11.59**	37.04
TATAAA	780	34.36	22.31	21.54	**15.64**	29.89	25.38	10.26	15.38	**6.67**	35.00
ACTAAA	690	43.48	28.41	39.42	**20.00**	29.59	37.97	20.29	21.16	**13.91**	31.43
AGTAAA	670	33.73	30.75	25.67	**20.60**	33.01	28.66	11.04	22.09	**8.06**	27.03
GATAAA	460	35.22	21.74	16.96	**10.43**	52.00	15.65	13.04	11.30	**8.70**	33.33
AATATA	410	31.22	23.90	25.85	**14.63**	38.78	27.80	7.80	11.71	**3.90**	50.00
CATAAA	410	40.98	22.93	27.80	**20.49**	10.64	24.39	14.63	16.59	**4.88**	66.67
AATAGA	370	22.16	6.49	9.73	**6.49**	0.00	25.95	9.73	20.00	**3.78**	61.11
Average	-	-	20.60	22.47	**16.26**	20.75	-	18.52	17.96	**12.59**	34.17

$D \times 2mL$, and the associated SVM model can be trained in the primal form (usually faster than in the dual form). At test time, PPK, SPE and WD need to compute the kernel values between each support vector (the number of support vectors can be large) and each test

data point. In contrast, our method only computes a feature vector of length $2mL$ for each data point, and then perform an inner product with the learned SVM model $\mathbf{w}$, a vector of length $2mL$.

Table 3.2: Runtime comparisons on two variants *AATAAA* and *ATTAAA* for one train/test split, with $k = 3$ and all other parameters set to optimal. PPK could not finish running within 48 hours on *AATAAA*.

Time (seconds)	*AATAAA*				*ATTAAA*			
	PPK	SPE	WD	HMM	PPK	SPE	WD	HMM
Training	-	46.16	37.38	**7.59**	2722.81	9.46	6.47	**3.86**
Testing	-	6.81	**0.94**	1.43	674.08	1.54	0.69	**0.67**

3.4.4 Comparison to state-of-the-art method: the random forest model

Table 3.3: Comparison of our method (HMM) with RF. The performance of both RF and HMM is evaluated on the same five-fold cross-validation. "Rel" denotes the relative improvement of HMM with respect to RF. The lowest value for each criterion of each motif variant is indicated in bold.

Variants	Size	Error Rate (%)			False Negative Rate (%)			False Positive Rate (%)		
		RF	HMM	Rel	RF	HMM	Rel	RF	HMM	Rel
AATAAA	5190	20.06	**18.59**	7.31	19.74	**18.54**	6.10	20.37	**18.65**	8.44
ATTAAA	2400	18.42	**16.21**	12.01	18.68	**18.17**	2.75	18.15	**14.25**	21.49
AAAAAG	1250	16.64	**9.36**	43.75	16.53	**11.36**	31.28	16.75	**7.36**	56.06
AAGAAA	1230	11.06	**5.45**	50.75	11.92	**6.02**	49.53	10.15	**4.88**	51.94
TATAAA	880	19.55	**15.34**	21.53	**18.10**	19.09	-5.47	20.87	**11.59**	44.46
AATACA	780	19.36	**11.15**	42.39	18.13	**15.64**	13.73	20.49	**6.67**	67.46
AGTAAA	690	27.83	**16.96**	39.07	25.24	**20.00**	20.76	29.92	**13.91**	53.50
ACTAAA	670	22.09	**14.33**	35.14	20.69	**20.60**	0.45	23.36	**8.06**	65.50
GATAAA	460	20.00	**9.57**	52.17	21.01	**10.43**	50.33	18.92	**8.70**	54.04
CATAAA	410	18.54	**9.27**	50.01	16.92	**14.63**	13.51	20.00	**3.90**	80.49
AATATA	410	24.88	**12.68**	49.02	24.12	**20.49**	15.06	25.59	**4.88**	80.94
AATAGA	370	18.38	**5.14**	72.06	19.37	**6.49**	66.51	17.32	**3.78**	78.15
Average	-	19.19	**14.42**	25.62	18.83	**16.26**	14.81	19.48	**12.59**	35.40

Table 3.3 compares the proposed method with a state-of-the-art by five-fold cross-validation on the 12 variants of human poly(A) motifs: the random forest (RF) model using domain-specific features by [39]. As shown in Table 3.3, our method is always significantly more accurate than RF (much lower error rates) and is more sensitive than RF on 11 out

of the 12 variants. In fact, our method improves the error rates by 7%-72% as compared to RF on the 12 motif variants. On average, our method has an improvement over RF in terms of the error rate, false positive rate and false negative rate by about 26%, 15% and 35%, respectively. These percentages imply that the significant improvement on the error rate is not just the result of a better trade off between sensitivity and specificity, but it is the result of being a better method in both senses. By comparing the results in Tables 3.1 and 3.3, it can be seen that the RF model outperforms other string kernels (PPK, SPE and WD) in terms of accuracy for poly(A) motif prediction. Our method that systematically extracts spectral latent features significantly improves upon RF and other string kernels on the same task. This supports our assumption that uncertainty and diversity information is important for this problem and there is a "clean" DNA signal hidden in the observed sequences.

3.4.5 Visualizing importance scores of dimers and positions

Another advantage of our method over previous state-of-the-art poly(A) motif predictors is that our method can be used to visualize the importance of k-mers or positions to the prediction task. This can provide researchers or users a direct and intuitive way to study the patterns and characteristics of DNA sequences. More importantly, most previous studies on poly(A) motifs try to reveal statistics of the surrounding regions of true motifs. Our method, in contrast, can reveal patterns for false motifs at the same time.

In Figure 3.1, we present the importance scores $\alpha(dimer@t)$ for dimers (subsequences of 2 nt, e.g., *AG*) in determining whether or not a candidate poly(A) motif is a true motif. A number of interesting observations can be made about this figure:

- *AA* is an informative subsequence to differentiate true poly(A) motifs from false ones in all 12 variants of human poly(A) motifs. When *AA* appears frequently within 30 nt downstream of the candidate poly(A) motif, this strongly suggests that it is a true poly(A) motif. This is expected because the mRNA cleavage site is often 15-25 nt downstream of the poly(A) motif, and the region between the cleavage site and the

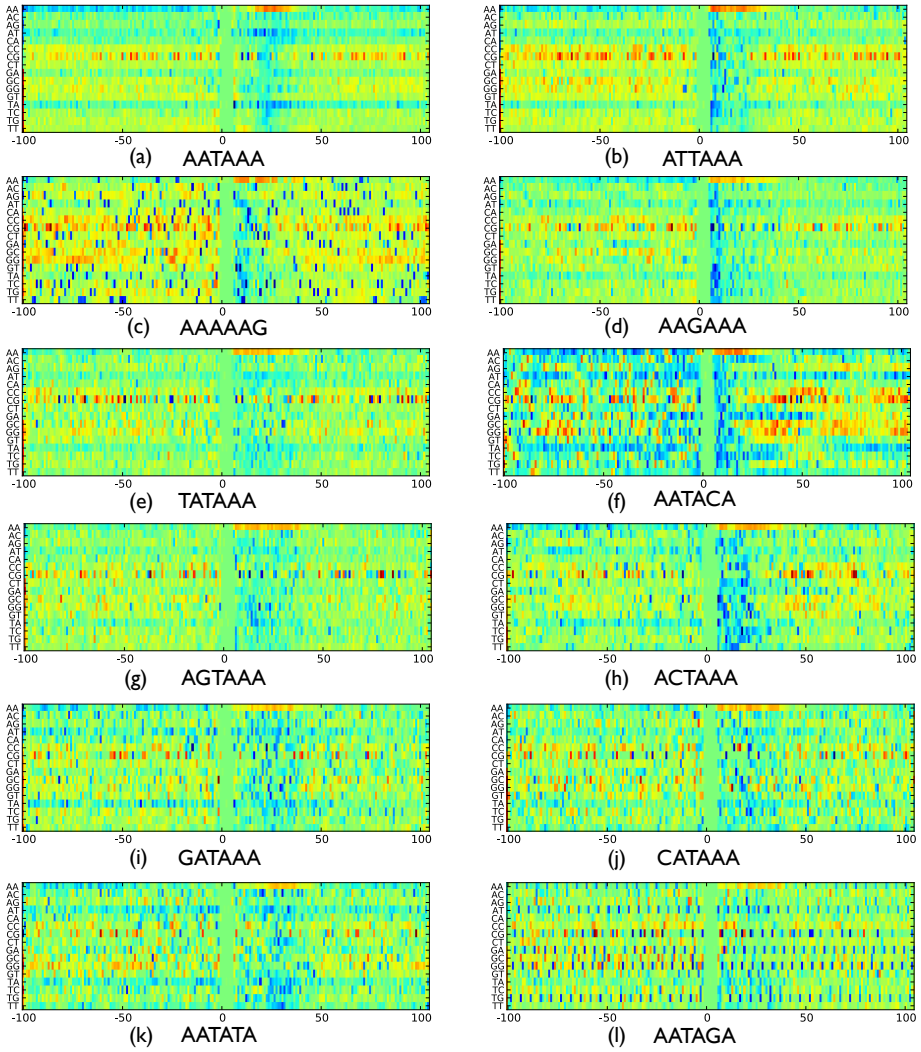


Figure 3.1: Visualization of the importance of different dimers at different positions for the 12 variants of human poly(A) motifs. The x-axis gives the positions in the sequence. The y-axis lists all 16 possible dimers. The colors denote the levels of importance: the light green color for the positions 0-6 is the background color, which indicates that no effects differentiate true and false motifs; the darker the red, the more important the dimer at that position is to identifying true motifs; the darker the blue, the more important the dimer at that position is to identifying false motifs.

motif is known to be *A*-rich [41]. Interestingly, when *AA* appears frequently within 100 nt upstream of the candidate motif, the candidate is likely to be a false motif.

- *CG* is an interesting dimer. Its positions carry important information for determining both true and false poly(A) motifs. When *CG* appears beyond 40 nt upstream of the motif, this suggests that the motif is a true poly(A) motif. On the other hand, *CG* serves as a strong sign for false predictions in the immediate 20 nt downstream and then becomes a sign for true predictions for further downstream sequences. In [51], it was found that -100/-41 and +41/+100 regions of poly(A) motifs are *C*- and *G*-rich regions. Our results suggest that looking at *CG* together rather than individually may capture more informative patterns.

- *TA* or *AT* is one of the characteristics for false poly(A) motifs among all 12 variants. No matter if it appears frequently in downstream or upstream nt sequences of the candidate poly(A) motif, *TA* or *AT* suggests that the candidate is a false one. Between them, *TA* is more informative than *AT* to specify false motifs. On the one hand, our findings coincide with previous studies that *TA* and *AT* are important features around the poly(A) motifs and *TA* is more frequent than *AT* (e.g., the *TATATA* oligonucleotide is more over-represented than the *ATATAT* oligonucleotide [40, 51]). On the other hand, our findings reveal that *TA* and *AT* appear much more often in sequences around the false motifs than around the true motifs. In [40, 51], it was found that *TATATA* and *ATATAT* are the most over-represented oligonucleotides downstream of the poly(A) motifs. However, by analyzing the negative motifs, our results imply that although *TA* and *AT* appear often in positive motifs, they appear even more often in negative ones.

- *TG* can determine false motifs at alternate positions upstream or downstream of the candidate motif of *AATAGA* (Figure 3.1(l)). This is not the case for the two most frequent motifs, *AATAAA* and *ATTAAA*, which partially supports our hypothesis that

the intrinsic characteristics of the frequent motifs and rare motifs are different. Thus, a good poly(A) motif predictor should have different models for different motif variants.

Figure 3.2 shows the importance scores for different positions with k from 1 to 5. Again, the 6 nt positions for the candidate motifs offer no information to the prediction. However, because the k-mers overlapping with the motif regions contain subsequences of the motifs, the motif regions do not have absolutely "zero" importance. Again, we list key observations here:

- The longer the subsequences are (bigger k), the smoother the importance curves are. This is expected because considering more nt at the same time will average the effects caused by individual positions.

- In almost all the variants, the 50 nt downstream of the candidate motifs are informative. Specifically, motif variants *AATAAA*, *CATAAA*, and *AATATA* have important information at the positions around the 25th nt downstream of the candidate motifs (Figures 3.2(a), (j) and (k)). This coincides with the fact that the mRNA cleavage site is 15-25 nt downstream of poly(A) motifs [40, 41].

3.5 Summary

In this chapter, we proposed a novel method to extract features from upstream and downstream regions of candidate poly(A) motifs in human DNA sequences. Our proposed spectral latent feature-based method achieves state-of-the-art results. The proposed method systematically explores the information encoded in nucleotide sequences by learning sequence dynamics and matching latent distributions on each position, and it can be easily extended to visualize the importance of subsequences and positions, thus providing a general method for sequence-based classification problems in bioinformatics. Our method can be directly applied to other sequence classification problems and achieve state-of-the-art results, such

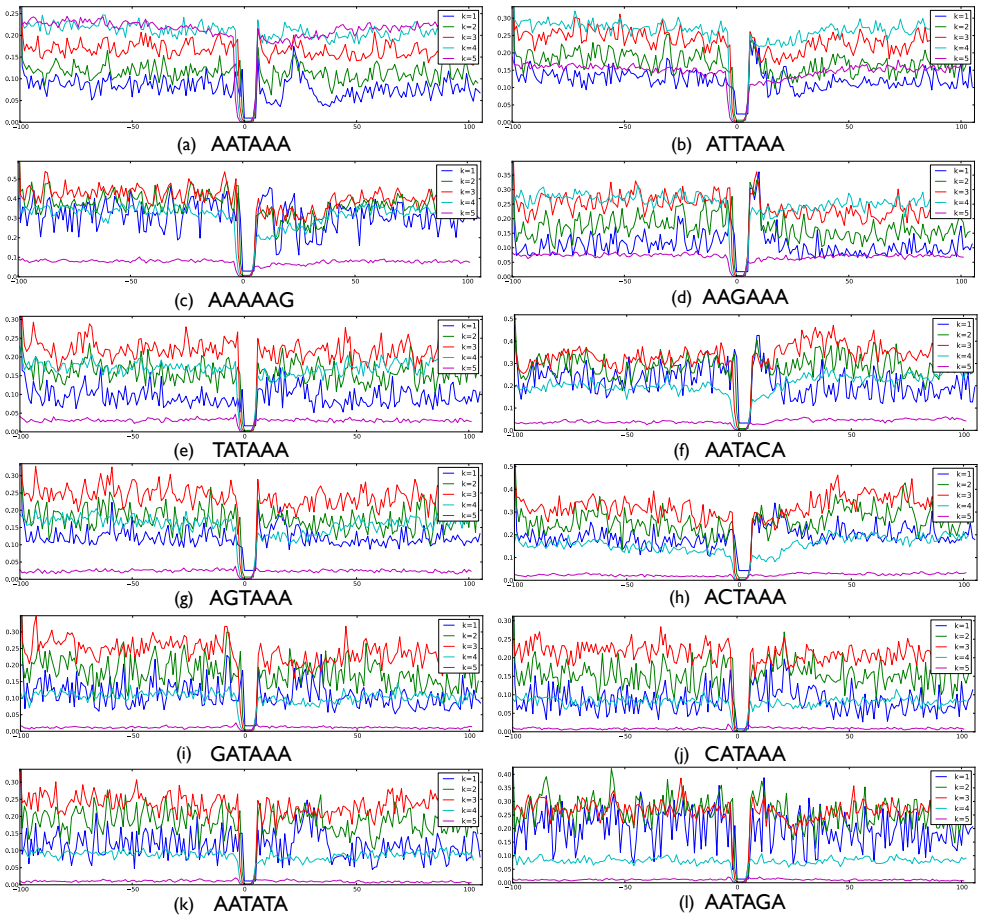


Figure 3.2: Visualization of the importance of different positions for the 12 motif variants. The x-axis gives the position in the sequence. For each k from 1 to 5, the y-axis is the importance score of a position by summing over the absolute values of the importance for all possible k-mers at that position.

as the transcription start site prediction and splice site prediction (Supplementary Material S1 and Table S1).

Our method, currently, requires a fixed length of upstream and downstream sequences for the training and testing data. Such prior knowledge has to be given as input. We are trying to generalize and extend our method to take varying lengths of sequences for different samples. Furthermore, there may be longer-range dependency between the latent variables, and hidden Markov models of higher orders may be needed for the feature extraction purpose, for which junction tree-type algorithms can be applied [52]. Similar to the weighted degree kernel with shifts [53], our method can also be straightforwardly extended to take shifted matches into account.

PART II: SCALABLE ALGORITHMS FOR NON-CONVEX OPTIMIZATION

The nonparametric spectral algorithm introduced in Chapter 2 does not scale to large datasets due to the quadratic computational complexity. However, the nonparametric method requires a large amount of data points to really demonstrate its advantage over alternatives. This calls for efficient algorithms to solve the key non-convex optimization problem, which boils down to kernel principle component analysis (KPCA). In this part, we describe two versions of scalable algorithms for KPCA.

In Chapter 4, we introduce the first version, which is a stochastic optimization algorithm that approximates the expectation with a few samples. Unlike traditional stochastic gradient descent (SGD), it needs to make two stochastic approximations: one for the expectation in terms of data points and another for the kernel function. In every iteration, the algorithm adaptively grows the model complexity after seeing more data points. Therefore, it achieves a better balance between computation and statistics. We show that it can solve KPCA on datasets with millions of data points.

In Chapter 5, we describe a distributed version of KPCA, where the computation is distributed among several workers. The algorithm approximately computes the leverage scores and then samples a few representative data points for the whole dataset. The key balance in this context is the tradeoff between approximation error and the communication overhead. We show the proposed algorithm has nearly optimal communication efficiency and can successfully run on datasets with millions of data points.

CHAPTER 4
DOUBLY STOCHASTIC GRADIENT FOR NONLINEAR COMPONENT ANALYSIS

Nonlinear component analysis such as kernel Principle Component Analysis (KPCA) is a key computation in spectral methods, but it can not scale up to big datasets. Recent attempts have employed random feature approximations to convert the problem to the primal form for linear computational complexity. However, to obtain high quality solutions, the number of random features should be the same order of magnitude as the number of data points, making such approach not directly applicable to the regime with millions of data points.

We propose a simple, computationally efficient, and memory friendly algorithm based on the "doubly stochastic gradients" to scale up a range of kernel nonlinear component analysis, such as kernel PCA, CCA and SVD. Despite the *non-convex* nature of these problems, our method enjoys theoretical guarantees that it converges at the rate $\tilde{O}(1/t)$ to the global optimum, even for the top k eigen subspace. Unlike many alternatives, our algorithm does not require explicit orthogonalization, which is infeasible on big datasets. We demonstrate the effectiveness and scalability of our algorithm on large scale synthetic and real world datasets.

4.1 Introduction

Scaling up nonlinear component analysis has been challenging due to prohibitive computation and memory requirements. Recently, methods such as Randomized Component Analysis [54] are able to scale to larger datasets by leveraging random feature approximation. Such methods approximate the kernel function by using explicit random feature mappings, then perform subsequent steps in the primal form, resulting in linear computational complexity. Nonetheless, theoretical analysis [55, 54] shows that in order to get high

quality results, the number of random features should grow linearly with the number of data points. Experimentally, one often sees that the statistical performance of the algorithm improves as one increases the number of random features.

Another approach to scale up the kernel component analysis is to use stochastic gradient descent and online updates [56]. These stochastic methods have also been extended to the kernel case [57, 58, 59]. They require much less computation than their batch counterpart, converge in $O(1/t)$ rate, and are naturally applicable to streaming data setting. Despite that, they share a severe drawback: all data points used in the updates need to be saved, rendering them impractical for large datasets.

In this chapter, we propose to use the "doubly stochastic gradients" for nonlinear component analysis. This technique is a general framework for scaling up kernel methods [60] for *convex problems* and has been successfully applied to many popular kernel machines such as kernel SVM, kernel ridge regressions, and Gaussian process. It uses two types of stochastic approximation simultaneously: random data points instead of the whole dataset (as in stochastic update rules), and random features instead of the true kernel functions (as in randomized component analysis). These two approximations lead to the following benefits:

- **Computation efficiency** The key computation is the generation of a mini-batch of random features and the evaluation of them on a mini-batch of data points, which is very efficient.

- **Memory efficiency** Instead of storing training data points, we just keep a small program for regenerating the random features, and sample previously used random features according to pre-specified random seeds. This leads to huge savings: the memory requirement up to step t is $O(t)$, independent of the dimension of the data.

- **Adaptibility** Unlike other approaches that can only work with a fixed number of random features beforehand, doubly stochastic approach is able to increase the model

complexity by using more features when new data points arrive, and thus enjoys the advantage of nonparametric methods.

Although on first look our method appears similar to the approach in [60], the two methods are fundamentally different. In [60], they address *convex problems*, whereas our problem is highly *non-convex*. The convergence result in [60] crucially relies on the properties of convex functions, which do not translate to our problem. Instead, our analysis centers around the stochastic update of power iterations, which uses a different set of proof techniques.

Since kernel PCA is a typical task, we focus on it in the main text and provide a description of other tasks in Section 4.6. Although we only state the guarantee for kernel PCA, the analysis naturally carries over to the other tasks.

4.2 Related work

Many efforts have been devoted to scale up kernel methods. The random feature approach [61, 55] approximates the kernel function with explicit random feature mappings and solves the problem in primal form, thus circumventing the quadratic computational complexity. It has been applied to various kernel methods [62, 60, 54], among which most related to our work is Randomized Component Analysis [54]. One drawback of Randomized Component Analysis is that their theoretical guarantees are only for kernel matrix approximation: it does not say anything about how close the solution obtained from randomized PCA is to the true solution. In contrast, we provide a finite time convergence rate of how our solution approaches the true solution. In addition, even though a moderate size of random features can work well for tens of thousands of data points, datasets with tens of millions of data points require many more random features. Our online approach allows the number of random features, hence the flexibility of the function class, to grow with the number of data points. This makes our method suitable for data streaming setting, which is not possible for previous approaches.

Online algorithms for PCA have a long history. Oja proposed two stochastic update rules for approximating the first eigenvector and provided convergence proof in [56], respectively. These rules have been extended to the generalized Hebbian update rules [63, 64, 65] that compute the top k eigenvectors (the subspace case). Similar ones have also been derived from the perspective of optimization and stochastic gradient descent [64, 66]. They are further generalized to the kernel case [57, 58, 59]. However, online kernel PCA needs to store all the training data, which is impractical for large datasets. Our doubly stochastic method avoids this problem by using random features and keeping only a small program for regenerating previously used random features according to pre-specified seeds. As a result, it can scale up to tens of millions of data points.

For finite time convergence rate, [65] proved the $O(1/t)$ rate for the top eigenvector in linear PCA using Oja's rule. For the same task, [67] proposed a noise reduced PCA with linear convergence rate, where the rate is in terms of epochs, *i.e.*, number of passes over the whole dataset. The noisy power method presented in [68] provided linear convergence for a subspace, although it only converges linearly to a constant error level. In addition, the updates require explicit orthogonalization, which is impractical for kernel methods. In comparison, our method converges in $O(1/t)$ for a subspace, without the need for orthogonalization.

4.3 Preliminaries

4.3.1 Covariance Operators

Given a distribution $\mathbb{P}(x)$, a kernel function $k(x, x')$ with RKHS $\mathcal{F}$, the covariance operator $A : \mathcal{F} \mapsto \mathcal{F}$ is a linear self-adjoint operator defined as

$$Af(\cdot) := \mathbb{E}_x[f(x)\, k(x, \cdot)], \quad \forall f \in \mathcal{F}, \tag{4.1}$$

and furthermore $\langle g, Af \rangle_{\mathcal{F}} = \mathbb{E}_x[f(x)\, g(x)], \forall g \in \mathcal{F}$.

Let $F = (f_1(\cdot), f_2(\cdot), \ldots, f_k(\cdot))$ be a list of k functions in the RKHS, and we define matrix-like notation

$$AF(\cdot) := (Af_1(\cdot), \ldots, Af_k(\cdot)), \tag{4.2}$$

and $F^\top AF$ is a $k \times k$ matrix, whose (i, j)-th element is $\langle f_i, Af_j \rangle_{\mathcal{F}}$. The outer-product of a function $v \in \mathcal{F}$ defines a linear operator $vv^\top : \mathcal{F} \mapsto \mathcal{F}$ such that

$$(vv^\top) f(\cdot) := \langle v, f \rangle_{\mathcal{F}} v(\cdot), \quad \forall f \in \mathcal{F} \tag{4.3}$$

Let $V = (v_1(\cdot), \ldots, v_k(\cdot))$ be a list of k functions, then the weighted sum of a set of linear operators, $\left\{v_i v_i^\top\right\}_{i=1}^k$, can be denoted using matrix-like notation as

$$V \Sigma_k V^\top := \sum_{i=1}^{k} \lambda_i v_i v_i^\top \tag{4.4}$$

where Σ_k is a diagonal matrix with λ_i on the i-th entry of the diagonal.

4.3.2 Kernel PCA

Kernel PCA aims to identify the top k eigenfunctions $V = (v_1(\cdot), \ldots, v_k(\cdot))$ for the covariance operator A, where V is also called the top k subspace for A.

A function v is an eigenfunction of covariance operator A with the corresponding eigenvalue λ if

$$Av(\cdot) = \lambda v(\cdot). \tag{4.5}$$

Given a set of eigenfunctions $\{v_i\}$ and associated eigenvalues $\{\lambda_i\}$, where $\langle v_i, v_j \rangle_{\mathcal{F}} = \delta_{ij}$.

We can denote the eigenvalue of A as

$$A = V\Sigma_k V^\top + V_\perp \Sigma_\perp V_\perp^\top \tag{4.6}$$

where $V = (v_1(\cdot), \ldots, v_k(\cdot))$ is the top k eigenfunctions of A, and Σ_k is a diagonal matrix with the corresponding eigenvalues, $V_\perp$ is the collection of the rest of the eigenfunctions, and $\Sigma_\perp$ is a diagonal matrix with the rest of the eigenvalues.

In the finite data case, the empirical covariance operator is $A = \frac{1}{n}\sum_i k(x_i, \cdot)k(x_i, \cdot)^\top$ or denoted as $\frac{1}{n}\sum_i k(x_i, \cdot) \otimes k(x_i, \cdot)$. According to the representer theorem, the solutions of the top k eigenfunctions of A can be expressed as linear combinations of the training points with the set of coefficients $\{\alpha_i\}_{i=1}^k \in \mathbb{R}^n$,

$$v_i = \sum_{j=1}^{n} \alpha_i^j k(x_j, \cdot)$$

Using $Av(\cdot) = \lambda v(\cdot)$ and the kernel trick, we have

$$K\alpha_i = \lambda_i \alpha_i,$$

where K is the $n \times n$ Gram matrix.

The infinite dimensional problem is thus reduced to a finite dimensional eigenvalue problem. However, this dual approach is clearly impractical on large scale datasets due quadratic memory and computational costs.

4.3.3 Random feature approximation

The usage of random features to approximate a kernel function is motivated by the following theorem.

Theorem 3 (Bochner) *A continuous, real-valued, symmetric and shift-invariant function $k(x - x')$ on $\mathbb{R}^d$ is a PD kernel if and only if there is a finite non-negative measure $\mathbb{P}(\omega)$ on*

Table 4.1: Summary of kernels in [61, 69, 70, 71, 72, 73, 74] and their explicit features

Kernel	$k(x, x')$	$\phi_\omega(x)$	$p(\omega)$
Gaussian	$\exp(-\frac{\|x-x'\|_2^2}{2})$	$\exp(-i\omega^\top x)$	$(2\pi)^{-\frac{d}{2}} \exp(-\frac{\|\omega\|_2^2}{2})$
Laplacian	$\exp(-\|x - x'\|_1)$	$\exp(-i\omega^\top x)$	$\prod_{i=1}^d \frac{1}{\pi(1+\omega_i^2)}$
Cauchy	$\prod_{i=1}^d \frac{2}{1+(x_i-x_i')^2}$	$\exp(-i\omega^\top x)$	$\exp(-\|\omega\|_1)$
Matérn	$\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|x-x'\|_2}{\ell}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}\|x-x'\|_2}{\ell}\right)$	$\exp(-i\omega^\top x)$	$h(\nu, d, \ell)\left(\frac{2\nu}{\ell^2}+4\pi^2\|\omega\|_2^2\right)^{\nu+d/2}$
Dot Product	$\sum_{n=0}^\infty a_n \langle x, x'\rangle^n \quad a_n \geq 0$	$\sqrt{a_N p^{N+1}} \prod_{i=1}^N \omega_i^\top x$	$\mathbb{P}[N = n] = \frac{1}{p^{n+1}}$
Polynomial	$(\langle x, x'\rangle + c)^p$	$\texttt{FFT}^{-1}(\texttt{FFT}(C_1 x) \odot \ldots \odot \texttt{FFT}(C_p x))$	$p(\omega_i^j \mid N = n) = \frac{1}{2}\frac{\omega_i^j+1}{2}\frac{1}{2}\frac{1-\omega_i^j}{2}$ $C_j = S_j D_j$ $D_j \in \mathbb{R}^{d\times d}\ S_j \in \mathbb{R}^{D\times d}$
Hellinger	$\sum_{i=1}^d \sqrt{x_i x_i'}$	$2\omega^\top \sqrt{x}$	$\frac{1}{2}\frac{\omega_i+1}{2}\frac{1}{2}\frac{1-\omega_i}{2},\ \omega_i \in \{-1, +1\}$
χ^2	$2\sum_{i=1}^d \frac{x_i x_i'}{x_i+x_i'}$	$\left[\exp(-i\omega \log x_j)\sqrt{x_j}\right]_{j=1}^d$	$sech(\pi\omega)$
Intersection	$\sum_{i=1}^d \min(x_i, x_i')$	$\left[\exp(-i\omega \log x_j)\sqrt{2x_j}\right]_{j=1}^d$	$\frac{1}{\pi(1+4\omega^2)}$
Jensen-Shannon	$\sum_{i=1}^d K_{JS}(x_i, x_i')$	$\left[\exp(-i\omega \log x_j)\sqrt{2x_j}\right]_{j=1}^d$	$\frac{sech(\pi\omega)}{\log 4(1+4\omega^2)}$
Skewed-χ^2	$2\prod_{i=1}^d \frac{\sqrt{x_i+c}\sqrt{x_i'+c}}{x_i+x_i'+2c}$	$\exp(-i\omega^\top \log(x + c))$	$\prod_{i=1}^d sech(\pi\omega_i)$
Skewed-Intersection	$\prod_{i=1}^d \min\left(\sqrt{\frac{x_i+c}{x_i'+c}}, \sqrt{\frac{x_i'+c}{x_i+c}}\right)$	$\exp(-i\omega^\top \log(x + c))$	$\prod_{i=1}^d \frac{1}{\pi(1+4\omega_i^2)}$
Exponential-Semigroup	$\exp(-\beta\sum_{i=1}^d \sqrt{x_i + x_j'})$	$\exp(-\omega^\top x)$	$\prod_{i=1}^d \frac{\beta}{2\sqrt{\pi}}\omega_i^{-\frac{3}{2}} \exp(-\frac{\beta}{4\omega_i})$
Reciprocal-Semigroup	$\prod_{i=1}^d \frac{\lambda}{x_i+x_i'+\lambda}$	$\exp(-\omega^\top x)$	$\prod_{i=1}^d \lambda \exp(-\lambda\omega_i)$
Arc-Cosine	$\frac{1}{\pi}\|x\|^n\|x'\|^n J_n(\theta)$	$(\omega^\top x)^n \max(0, \omega^\top x)$	$2\pi^{-\frac{d}{2}} \exp(-\frac{\|\omega\|_2^2}{2})$

D_j is random $\{\pm 1\}$ diagonal matrix and the columns of S_j are uniformly selected from $\{e_1, \ldots, e_D\}$. ν and ℓ are positive parameters. $h(\nu, d, \ell) = \frac{2^d \pi^{d/2}\Gamma(\nu+d/2)(2\nu)^\nu}{\Gamma(\nu)\ell^{2\nu}}$. K_ν is a modified Bessel function. $K_{JS}(x, x') = \frac{x}{2}\log_2 \frac{x+x'}{x} + \frac{x'}{2}\log_2 \frac{x+x'}{x'}$.

$$\theta = \cos^{-1}\frac{x^\top x'}{\|x\|\|x'\|},\ J_n(\theta) = (-1)^n(\sin\theta)^{2n+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n\left(\frac{\pi-\theta}{\sin\theta}\right)$$

$\mathbb{R}^d$, *such that* $k(x - x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')}\, d\mathbb{P}(\omega) = \int_{\mathbb{R}^d \times [0,2\pi]} \phi_\omega(x)\phi_\omega(y)\, d\left(\mathbb{P}(\omega) \times \mathbb{P}(b)\right)$, *where* $\mathbb{P}(b)$ *is a uniform distribution on* $[0, 2\pi]$, *and* $\phi_\omega(x) = \sqrt{2}\cos(\omega^\top x + b)$.

The theorem says that any shift-invariant kernel function $k(x, y) = k(x - y)$, *e.g.*, Gaussian RBF kernel, can be considered as an expectation of two feature functions $\phi_\omega(x)$ and $\phi_\omega(y)$, where the expectation is taked over a distribution on the random frequency ω and phase b.

We can therefore approximate the kernel function as an empirical average of samples from the distribution. In other words,

$$k(x, y) \approx \frac{1}{B} \sum_i \phi_{\omega_i}(x)\phi_{\omega_i}(y),$$

where $\{(\omega_i, b_i)\}_i^B$ are i.i.d. samples drawn from from $\mathbb{P}(\omega)$ and $\mathbb{P}(b)$, respectively.

The specific random feature functions and distributions have been worked out for many popular kernels. For Gaussian RBF kernel, $k(x-x') = \exp(-\|x-x'\|^2/2\sigma^2)$, this yields a Gaussian distribution $\mathbb{P}(\omega)$ with density proportional to $\exp(-\sigma^2\|\omega\|^2/2)$; for the Laplace kernel, this yields a Cauchy distribution; and for the Martern kernel, this yields the convolutions of the unit ball [75]. Similar representation where the explicit form of $\phi_\omega(x)$ and $\mathbb{P}(\omega)$ are known can also be derived for rotation invariant kernel, $k(x, x') = k(\langle x, x'\rangle)$, using Fourier transformation on sphere [75]. For polynomial kernels, $k(x, x') = (\langle x, x'\rangle + c)^p$, a random tensor sketching approach can also be used [71]. See Table 4.1 for explicit representations of different kernels.

4.4 Algorithm

In this section, we describe an efficient algorithm based on the "doubly stochastic gradients" to scale up kernel PCA. KPCA is essentially an eigenvalue problem in a functional space. Traditional approaches convert it to the dual form, leading to another eigenvalue problem whose size equals the number of training points, which is not scalable. Other ap-

proaches solve it in the primal form with stochastic functional gradient descent. However, these algorithms need to store all the training points seen so far. They quickly run into memory issues when working with hundreds of thousands of data points.

We propose to tackle the problem with "doubly stochastic gradients", in which we make two unbiased stochastic approximations. One stochasticity comes from sampling data points as in stochastic gradient descent. Another source of stochasticity is from random features to approximate the kernel.

One technical difficulty in designing doubly stochastic KPCA is an explicit orthogonalization step required in the update rules, which ensures the top k eigenfunctions are orthogonal. This is infeasible for kernel methods on a large dataset since it requires solving an increasingly larger KPCA problem in every iteration. To solve this problem, we formulate the orthogonality constraints into Lagrange multipliers which leads to an Oja-style update rule. The new update enjoys small per iteration complexity and converges to the ground-truth subspace.

We present the algorithm by first deriving the stochastic functional gradient update without random feature approximations, then introducing the doubly stochastic updates.

4.4.1 Stochastic functional gradient update

Kernel PCA can be formulated as the following *non-convex* optimization problem

$$\max_{G} \operatorname{tr}\left(G^\top A G\right) \quad \text{s.t.}\, G^\top G = I, \tag{4.7}$$

where $G := \left(g^1, \ldots, g^k\right)$ and g^i is the i-th function.

The Lagrangian that incorporates the constraint is

$$L(G, \Lambda) = \operatorname{tr}\left(G^\top A G\right) + \operatorname{tr}\left(\left(G^\top G - I\right)\Lambda\right)$$

where Λ is the Lagrangian multiplier. The gradient of the Lagrangian w.r.t G is

$$\nabla_G L = 2AG + G\left(\Lambda + \Lambda^\top\right).$$

Furthermore, from the optimality conditions

$$\begin{aligned} 2AG + G\left(\Lambda + \Lambda^\top\right) &= 0, \\ G^\top G - I &= 0, \end{aligned}$$

we can find $\Lambda + \Lambda^\top = -2G^\top AG$.

Plugging this into the gradient, it suggests the following update rule

$$G_{t+1} = G_t + \eta_t \left(I - G_t G_t^\top\right) AG_t. \tag{4.8}$$

Using a stochastic approximation for A: $A_t f(\cdot) = f(x_t)\, k(x_t, \cdot)$, we have $A_t G_t = k(x_t, \cdot) g_t^\top$ and $G_t^\top A_t G_t = g_t g_t^\top$, where $g_t = \left[g_t^1(x_t), \ldots, g_t^k(x_t)\right]^\top$. Therefore, the update rule is

$$G_{t+1} = G_t \left(I - \eta_t g_t g_t^\top\right) + \eta_t k(x_t, \cdot) g_t^\top. \tag{4.9}$$

This rule can also be derived using stochastic gradient and Oja's rule [56].

4.4.2 Doubly stochastic update

The update rule (4.9) has a fundamental computational drawback. At each time step t, a new basis $k(x_t, \cdot)$ is added to G_t, and it is therefore a linear combination of the feature mappings of all the data points up to t. This requires the algorithm to store all the data points it has seen so far, which is impractical for large scale datasets.

To address this issue, we use the random feature approximation $k(x, \cdot) \approx \phi_{\omega_i}(x)\phi_{\omega_i}(\cdot)$.

Algorithm 1: $\{\alpha_i\}_1^t = \textbf{DSGD-KPCA}(\mathbb{P}(x), k)$

Require: $\mathbb{P}(\omega)$, $\phi_\omega(x)$.

1: **for** $i = 1, \ldots, t$ **do**
2: Sample $x_i \sim \mathbb{P}(x)$.
3: Sample $\omega_i \sim \mathbb{P}(\omega)$ with seed i.
4: $h_i = \textbf{Evaluate}(x_i, \{\alpha_j\}_{j=1}^{i-1}) \in \mathbb{R}^k$.
5: $\alpha_i = \eta_i \phi_{\omega_i}(x_i) h_i$.
6: $\alpha_j = \alpha_j - \eta_i \alpha_j^\top h_i h_i$, for $j = 1, \ldots, i-1$.
7: **end for**

Denote H_t the function we get at iteration t, the update rule becomes

$$H_{t+1} = H_t \left(I - \eta_t h_t {h_t}^\top\right) + \eta_t \phi_{\omega_t}(x_t) \phi_{\omega_t}(\cdot) {h_t}^\top, \tag{4.10}$$

where h_t is the evaluation of H_t at the current data point: $h_t = \left[h_t^1(x_t), \ldots, h_t^k(x_t)\right]^\top$.

Given $H_0 = V_0$, we can explicitly represent H_t as a linear combination of all the random feature functions $\phi_{\omega_i}(\cdot)$:

$$H_t = \sum_i \phi_{\omega_i}(\cdot) \alpha_i^\top + V_0 \beta,$$

where $\alpha_i \in \mathbb{R}^k$ are the coefficients, and $\beta = \prod_{i \le t} \left(I - \eta_i h_i {h_i}^\top\right)$.

The update rule on the functions corresponds to the following update for the coefficients

$$\alpha_{t+1} = \eta_t \phi_{\omega_t}(x_t) h_t$$
$$\alpha_i = \alpha_i - \eta_t \alpha_i^\top h_t h_t, \quad \forall i \le t$$

The specific updates in terms of the coefficients are summarized in Algorithms 1 and 2. Note that in theory new random features are drawn in each iteration, but in practice one can revisit these random features.

Algorithm 2: $h =$ **Evaluate**$(x,\ \{\alpha_i\}_{i=1}^t)$

Require: $\mathbb{P}(\omega),\ \phi_\omega(x)$.

Set $h = 0 \in \mathbb{R}^k$.
for $i = 1, \ldots, t$ **do**
    Sample $\omega_i \sim \mathbb{P}(\omega)$ with seed i.
    $h = h + \phi_{\omega_i}(x)\alpha_i$.
end for

4.5 Analysis

In this section, we provide finite time convergence guarantees for our algorithm. As discussed in the previous section, explicit orthogonalization is not scalable for the kernel case, therefore we need to provide guarantees for the updates without orthogonalization. This challenge is even more prominent when using random features, since it introduces additional variance.

Furthermore, our guarantees are w.r.t. the top k-dimension subspace. Although the convergence without normalization for a top eigenvector has been established before [56], the subspace case is complicated by the fact that there are k angles between k-dimension subspaces, and we need to bound the *largest* angle. To the best of our knowledge, our result is the first finite time convergence result for a subspace *without* explicit orthogonalization.

Note that even though it appears our algorithm is similar to [60] on the surface, the underlying analysis is fundamentally different. In [60], the result only applies to *convex problems* where every local optimum is a global optimum while the problems we consider are highly *non-convex*. As a result, many techniques that [60] builds upon are not applicable.

4.5.1 Notations

In order to analyze the convergence of our doubly stochastic kernel PCA algorithm, we will need to define a few intermediate subspaces. For simplicity of notation, we will assume the mini-batch size for the data points is one.

1. Let $F_t := \left(f_t^1, \ldots, f_t^k\right)$ be the subspace estimated using stochastic gradient and explicit orthogonalization:

$$\tilde{F}_{t+1} \leftarrow F_t + \eta_t A_t F_t$$
$$F_{t+1} \leftarrow \tilde{F}_{t+1} \left(\tilde{F}_{t+1}^\top \tilde{F}_{t+1}\right)^{-1/2}$$

2. Let $G_t := \left(g_t^1, \ldots, g_t^k\right)$ be the subspace estimated using stochastic update rule without orthogonalization:

$$G_{t+1} \leftarrow G_t + \eta_t \left(I - G_t G_t^\top\right) A_t G_t.$$

where $A_t G_t$ and $G_t G_t^\top A_t G_t$ can be equivalently written using the evaluation of the function $\{g_t^i\}$ on the current data point, leading to the equivalent rule :

$$G_{t+1} \leftarrow G_t \left(I - \eta_t g_t g_t^\top\right) + \eta_t k(x_t, \cdot) g_t^\top. \tag{4.11}$$

3. Let $\tilde{G}_t := \left(\tilde{g}_t^1, \ldots, \tilde{g}_t^k\right)$ be the subspace estimated using stochastic update rule without orthogonalization, but the evaluation of the function $\{\tilde{g}_t^i\}$ on the current data point is replaced by the evaluation $h_t = \left[h_t^i(x_t)\right]^\top$:

$$\tilde{G}_{t+1} \leftarrow \tilde{G}_t + \eta_t k(x_t, \cdot) h_t^\top - \eta_t \tilde{G}_t h_t h_t^\top$$

4. Let $H_t := \left(h_t^1, \ldots, h_t^k\right)$ be the subspace estimated using doubly stochastic update rule without orthogonalization, *i.e.*, the update rule:

$$H_{t+1} \leftarrow H_t + \eta_t \phi_{\omega_t}(x_t) \phi_{\omega_t}(\cdot) h_t^\top - \eta_t H_t h_t h_t^\top. \tag{4.12}$$

The relation of these subspaces are summarized in Table 4.2. Using these notations, we describe a sketch of our analysis in the rest of the section, while the complete proofs are

Table 4.2: Relation between various subspaces.

Subspace	Evaluation	Orth.	Data Mini-batch	RF Mini-batch
V	–	–	–	–
F_t	$f_t(x)$	✓	✓	✗
G_t	$g_t(x)$	✗	✓	✗
$\tilde{G}_t$	$h_t(x)$	✗	✓	✗
H_t	$h_t(x)$	✗	✓	✓

provided in the appendix.

We first consider the subspace G_t estimated using the stochastic update rule, since it is simpler and its proof can provide the bases for analyzing the subspace H_t estimated by the doubly stochastic update rule.

4.5.2 Conditions and Assumptions

We will focus on the case when a good initialization V_0 is given:

$$V_0^\top V_0 = I, \ \ \cos^2\theta(V, V_0) \geq 1/2. \tag{4.13}$$

In other words, we analyze the later stage of the convergence, which is typical in the literature (*e.g.*, [67]). The early stage can be analyzed using established techniques (*e.g.*, [65]).

Throughout this chapter we suppose $|k(x, x')| \leq \kappa, |\phi_\omega(x)| \leq \phi$ and regard κ and ϕ as constants. Note that this is true for all the kernels and corresponding random features considered. We further regard the eigengap $\lambda_k - \lambda_{k+1}$ as a constant, which is also true for typical applications and datasets.

4.5.3 Update without random features

Our guarantee is on the cosine of the principal angle between the computed subspace and the ground truth eigen subspace (also called potential function): $\cos^2\theta(V, G_t) = \min_w \frac{\left\|V^\top G_t w\right\|^2}{\|G_t w\|^2}$.

Consider the two different update rules, one with explicit orthogonalization and another without

$$F_{t+1} \leftarrow \textbf{orth}(F_t + \eta_t A_t F_t)$$
$$G_{t+1} \leftarrow G_t + \eta_t \left(I - G_t G_t^\top\right) A_t G_t$$

where A_t is the empirical covariance of a mini-batch. Our final guarantee for G_t is the following.

Theorem 4 *Assume (4.13) and suppose the mini-batch sizes satisfy that for any* $1 \leq i \leq t$, $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$. *There exist step sizes* $\eta_i = O(1/i)$ *such that*

$$1 - \cos^2 \theta(V, G_t) = O(1/t).$$

The convergence rate $O(1/t)$ is in the same order as that of computing only the top eigenvector in linear PCA [65]. The bound requires the mini-batch size is large enough so that the spectral norm of A is approximated up to the order of the eigengap. This is because the increase of the potential is in the order of the eigengap. Similar terms appear in the analysis of the noisy power method [68] which, however, requires orthogonalization and is not suitable for the kernel case. We do not specify the mini-batch size, but by assuming suitable data distributions, it is possible to obtain explicit bounds; see for example [76, 77].

Proof sketch We first prove the guarantee for the orthogonalized subspace F_t which is more convenient to analyze, and then show that the updates for F_t and G_t are first order equivalent so G_t enjoys the same guarantee. To do so, we will require lemma 5 and 6 below

Lemma 5 $1 - \cos^2 \theta(V, F_t) = O(1/t)$.

Let c_t^2 denote $\cos^2 \theta(V, F_t)$, then a key step in proving the lemma is to show the following

recurrence

$$c_{t+1}^2 \geq c_t^2(1 + 2\eta_t(\lambda_k - \lambda_{k+1} - 2\left\|A - A_t\right\|)(1 - c_t^2)) - O(\eta_t^2). \tag{4.14}$$

We will need the mini-batch size large enough so that $2\left\|A - A_t\right\|$ is smaller than the eigen-gap.

Another key element in the proof of the theorem is the first order equivalence of the two update rules. To show this, we introduce $F(G_t) \leftarrow \textbf{orth}(G_t + \eta_t A_t G_t)$ to denote the subspace by applying the update rule of F_t on G_t. We show that the potentials of G_{t+1} and $F(G_t)$ are close:

Lemma 6 $\cos^2\theta(V, G_{t+1}) = \cos^2\theta(V, F(G_t)) \pm O(\eta_t^2)$.

The lemma means that applying the two update rules to the same input will result in two subspaces with similar potentials. Then by (4.14), we have $1 - \cos^2\theta(V, G_t) = O(1/t)$ which leads to our theorem. The proof of Lemma 6 is based on the observation that $\cos^2\theta(V, X) = \lambda_{\min}(V^\top X(X^\top X)^{-1}X^\top V)$. Comparing the Taylor expansions w.r.t. η_t for $X = G_{t+1}$ and $X = F(G_t)$ leads to the lemma.

4.5.4 Doubly stochastic update

The H_t computed in the doubly stochastic update is no longer in the RKHS so the principal angle is not well defined. Instead, we will compare the evaluation of functions from H_t and the true principal subspace V respectively on a point x. Formally, we show that for any function $v \in V$ with unit norm $\left\|v\right\|_{\mathcal{F}} = 1$, there exists a function h in H_t such that for any x, err $:= |v(x) - h(x)|^2$ is small with high probability.

To do so, we need to introduce a companion update rule: $\tilde{G}_{t+1} \leftarrow \tilde{G}_t + \eta_t k(x_t, \cdot)h_t^\top - \eta_t \tilde{G}_t h_t h_t^\top$ resulting in function in the RKHS, but the update makes use of function values from $h_t \in H_t$ which outside the RKHS. Let $w = \tilde{G}^\top v$ be the coefficients of v projected

onto $\tilde{G}$, $h = H_t w$, and $\tilde{g} = \tilde{G}_t w$. Then the error can be decomposed as

$$|v(x) - h(x)|^2 = |v(x) - \tilde{g}(x) + \tilde{g}(x) - h(x)|^2 \leq 2\,|v(x) - \tilde{g}(x)|^2 + 2\,|\tilde{g}(x) - h(x)|^2$$
$$\leq \underbrace{2\kappa^2\,\|v - \tilde{g}\|_{\mathcal{F}}^2}_{\text{(I: Lemma 8)}} + \underbrace{2\,|\tilde{g}(x) - h(x)|^2}_{\text{(II: Lemma 9)}}. \tag{4.15}$$

By definition, $\|v - \tilde{g}\|_{\mathcal{F}}^2 = \|v\|_{\mathcal{F}}^2 - \|\tilde{g}\|_{\mathcal{F}}^2 \leq 1 - \cos^2\theta(V, \tilde{G}_t)$, so the first error term can be bounded by the guarantee on $\tilde{G}_t$, which can be obtained by similar arguments in Theorem 4. For the second term, note that $\tilde{G}_t$ is defined in such a way that the difference between $\tilde{g}(x)w$ and $h(x)$ is a martingale, which can be bounded by careful analysis.

Theorem 7 *Assume (4.13) and suppose the mini-batch sizes satisfy that for any* $1 \leq i \leq t$, $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$ *and are of order* $\Omega(\ln \frac{t}{\delta})$. *There exist step sizes* $\eta_i = O(1/i)$, *such that the following holds. If* $\Omega(1) = \lambda_k(\tilde{G}_i^\top \tilde{G}_i) \leq \lambda_1(\tilde{G}_i^\top \tilde{G}_i) = O(1)$ *for all* $1 \leq i \leq t$, *then for any* x *and any function* v *in the span of* V *with unit norm* $\|v\|_{\mathcal{F}} = 1$, *we have that with probability at least* $1 - \delta$, *there exists* h *in the span of* H_t *satisfying* $|v(x) - h(x)|^2 = O\left(\frac{1}{t}\ln\frac{t}{\delta}\right)$.

The point-wise error scales as $\tilde{O}(1/t)$ with the step t. Besides the condition that $\|A - A_i\|$ is up to the order of the eigengap, we additionally need that the random features approximate the kernel function up to constant accuracy on all the data points up to time t, which eventually leads to $\Omega(\ln \frac{t}{\delta})$ mini-batch sizes. Finally, we need $\tilde{G}_i^\top \tilde{G}_i$ to be roughly isotropic, *i.e.*, $\tilde{G}_i$ is roughly orthonormal. Intuitively, this should be true for the following reasons: $\tilde{G}_0$ is orthonormal; the update for $\tilde{G}_t$ is close to that for G_t, which in turn is close to F_t that are orthonormal.

Proof sketch In order to bound term I in (4.15), we show that

Lemma 8 $1 - \cos^2\theta(V, \tilde{G}_t) = O\left(\frac{1}{t}\ln\frac{t}{\delta}\right)$.

This is proved by following similar arguments to get the recurrence (4.14), except with an additional error term, which is caused by the fact that the update rule for $\tilde{G}_{t+1}$ is using the

evaluation $h_t(x_t)$ rather than $\tilde{g}_t(x_t)$. Bounding this additional term thus relies on bounding the difference between $h_t(x) - \tilde{g}_t(x)$, which is also what we need for bounding term II in (4.15). For this, we show:

Lemma 9 *For any x and unit vector w, with probability $\geq 1 - \delta$ over $(\mathcal{D}^t, \omega^t)$, $|\tilde{g}_t(x)w - h_t(x)w|^2 = O\left(\frac{1}{t}\ln\left(\frac{t}{\delta}\right)\right)$.*

The key to prove this lemma is that our construction of $\tilde{G}_t$ makes sure that the difference between $\tilde{g}_t(x)w$ and $h_t(x)w$ consists of their difference in each time step. Furthermore, the difference forms a martingale and thus can be bounded by Azuma's inequality. See the supplementary for the details.

4.6 Extensions

The proposed algorithm is a general technique for solving eigenvalue problems in the functional space. Numerous machine learning algorithms boil down to this fundamental operation. Therefore, our method can be easily extended to solve many related tasks, including latent variable estimation, kernel CCA, spectral clustering, *etc.*.

We briefly illustrate how to extend to different machine learning algorithms in the following subsections.

4.6.1 Locating individual eigenfunctions

The proposed algorithm finds the subspace spanned by the top k eigenfunctions, but it does not isolate the individual eigenfunctions. When we need to locate these individual eigenfunctions, we can use a modified version, called Generalized Hebbian Algorithm (GHA) [63]. Its update rule is

$$G_{t+1} = G_t + \eta_t A_t G_t - \eta_t G_t \,\mathrm{UT}\left[G_t^\top A_t G_t\right], \tag{4.16}$$

where $\mathrm{UT}[\cdot]$ is an operator that sets the lower triangular parts to zero.

To understand the effect of the upper triangular operator, we can see that $\mathrm{UT}[\cdot]$ forces the update rule for the first function of G_t to be exactly the same as that of one-dimensional subspace; all the contributions from the other functions are zeroed out.

$$g_{t+1}^1 = g_t^1 + \eta_t A_t g_t^1 - \eta_t g_t^1 {g_t^1}^\top A_t g_t^1, \tag{4.17}$$

Therefore, the first function will converge to the eigenfunction corresponding to the top eigenvalue.

For all the other functions, $\mathrm{UT}[\cdot]$ implements a Gram-Schmidt-like orthogonalization that subtracts the contributions from other eigenfunctions.

4.6.2 Latent variable models and kernel SVD

Latent variable models are probabilistic models that assume unobserved or latent structures in the data. It appears in specific forms such as Gaussian Mixture Models (GMM), Hidden Markov Models (HMM) and Latent Dirichlet Allocations (LDA), *etc.*.

The EM algorithm [4] is considered the standard approach to solve such models. Recently, spectral methods have been proposed to estimate latent variable models with provable guarantees [11, 78]. Compared with the EM algorithm , spectral methods are faster to compute and do not suffer from local optima.

The key algorithm behind spectral methods is the SVD. However, kernel SVD scales quadratically with the number of data points. Our algorithm can be straightforwardly extended to solve kernel SVD. The extension hinges on the following relation

$$\begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} V \\ U \end{bmatrix} = \begin{bmatrix} A^\top U \\ AV \end{bmatrix} = \begin{bmatrix} V \\ U \end{bmatrix} \Sigma,$$

where $U\Sigma V^\top$ is the SVD of A.

It is therefore reduced to the eigenvalue problem. Plugging it into the update rule and

Algorithm 3: $\{\alpha_i, \beta_i\}_1^t = \textbf{DSGD-KSVD}(\mathbb{P}(x), \mathbb{P}(y), k)$

Require: $\mathbb{P}(\omega)$, $\phi_\omega(x)$.

for $i = 1, \ldots, t$ **do**
  Sample $x_i \sim \mathbb{P}(x)$. Sample $y_i \sim \mathbb{P}(y)$.
  Sample $\omega_i \sim \mathbb{P}(\omega)$ with seed i.
  $u_i = \textbf{Evaluate}(x_i, \{\alpha_j\}_{j=1}^{i-1}) \in \mathbb{R}^k$.
  $v_i = \textbf{Evaluate}(y_i, \{\beta_j\}_{j=1}^{i-1}) \in \mathbb{R}^k$.
  $W = u_i v_i^\top + v_i u_i^\top$
  $\alpha_i = \eta_i \phi_{\omega_i}(x_i) v_i$.
  $\beta_i = \eta_i \phi_{\omega_i}(y_i) u_i$.
  $\alpha_j = \alpha_j - \eta_i W \alpha_j$, for $j = 1, \ldots, i-1$.
  $\beta_j = \beta_j - \eta_i W \beta_j$, for $j = 1, \ldots, i-1$.
end for

treating the two blocks separately, we thus get two simultaneous update rules

$$W_t = U_t^\top A V_t + V_t^\top A^\top U_t \tag{4.18}$$

$$U_{t+1} = U_t + \eta_t \left(A V_t - U_t W_t\right), \tag{4.19}$$

$$V_{t+1} = V_t + \eta_t \left(A^\top U_t - V_t W_t\right). \tag{4.20}$$

The algorithm for updating the coefficients is summarized in Algorithm 3.

4.6.3 Kernel CCA and generalized eigenvalue problem

Kernel CCA and ICA [79] can also be solved under the proposed framework because they can be viewed as generalized eigenvalue problem.

Given two variables X and Y, CCA finds two projections such that the correlations between the two projected variables are maximized. Given the covariance matrices C_{XX}, C_{YY}, and C_{XY}, CCA is equivalent to the following problem

$$\begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix} \begin{bmatrix} g_X \\ g_Y \end{bmatrix} = \left(1 + \sigma^2\right) \begin{bmatrix} C_{XX} & \\ & C_{YY} \end{bmatrix} \begin{bmatrix} g_X \\ g_Y \end{bmatrix},$$

where g_X and g_Y are the top canonical correlation functions for variables X and Y, respec-

tively, and σ is the corresponding canonical correlation.

This is a generalized eigenvalue problem. It can reformulated as the following non-convex optimization problem

$$\max_{G} \operatorname{tr}\left(G^{\top} A G\right), \tag{4.21}$$

$$\text{s.t. } G^{\top} B G = I. \tag{4.22}$$

Following the derivation for the standard eigenvalue problem, we get the foliowing update rules

$$G_{t+1} = G_t + \eta_t \left(I - B G_t G_t^{\top}\right) A G_t. \tag{4.23}$$

Denote G_t^X and G_t^Y the canonical correlation functions for X and Y, respectively. We can rewrite the above update rule as two simultaneous rules

$$W_t = {G_t^Y}^{\top} C_{YX} G_t^X + {G_t^X}^{\top} C_{XY} G_t^Y \tag{4.24}$$

$$G_{t+1}^X = G_t^X + \eta_t \left[C_{XY} G_t^Y - C_{XX} G_t^X W\right] \tag{4.25}$$

$$G_{t+1}^Y = G_t^Y + \eta_t \left[C_{YX} G_t^X - C_{YY} G_t^Y W\right]. \tag{4.26}$$

We present the detailed updates for coefficients in Algorithm 4.

4.6.4 Kernel sliced inverse regression

Kernel sliced inverse regression [80] aims to do sufficient dimension reduction in which the found low dimension representation preserves the statistical correlation with the targets. It also reduces to a generalized eigenvalue problem, and has been shown to find the same subspace as KCCA [80].

Algorithm 4: $\{\alpha_i, \beta_i\}_1^t = \textbf{DSGD-KCCA}(\mathbb{P}(x), \mathbb{P}(y), k)$

Require: $\mathbb{P}(\omega)$, $\phi_\omega(x)$.

for $i = 1, \ldots, t$ **do**
  Sample $x_i \sim \mathbb{P}(x)$. Sample $y_i \sim \mathbb{P}(y)$.
  Sample $\omega_i \sim \mathbb{P}(\omega)$ with seed i.
  $u_i = \textbf{Evaluate}(x_i, \{\alpha_j\}_{j=1}^{i-1}) \in \mathbb{R}^k$.
  $v_i = \textbf{Evaluate}(y_i, \{\beta_j\}_{j=1}^{i-1}) \in \mathbb{R}^k$.
  $W = u_i v_i^\top + v_i u_i^\top$
  $\alpha_i = \eta_i \phi_{\omega_i}(x_i)\,[v_i - W u_i]$.
  $\beta_i = \eta_i \phi_{\omega_i}(y_i)\,[u_i - W v_i]$.
end for

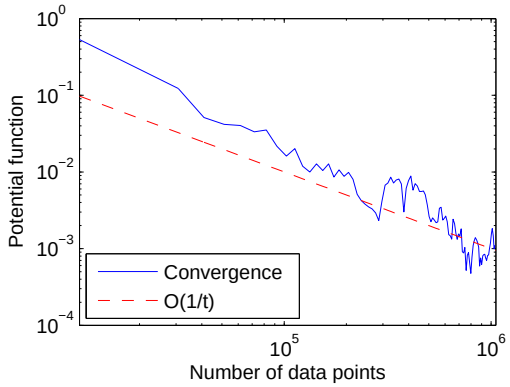


Figure 4.1: Convergence for DSGD-KPCA on the dataset with analytical solution.

4.7 Experiments

We demonstrate the effectiveness and scalability of our algorithm on both synthetic and real world datasets.

4.7.1 Synthetic dataset with analytical solution

We first verify the convergence rate of DSGD-KPCA on a synthetic dataset with analytical solution of eigenfunctions [81]. If the data follow a Gaussian distribution, and we use a Gaussian kernel, then the eigenfunctions are given by the Hermite polynomials.

We generated 1 million data points, and ran DSGD-KPCA with a total of 262,144 random features. In each iteration, we use a data mini-batch of size 512, and a random

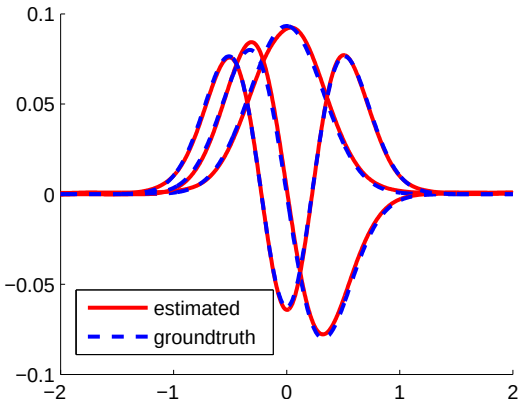


Figure 4.2: Recovered top 3 eigenfunctions using DSGD-KPCA on the dataset with analytical solution.

feature mini-batch of size 128. After all random features are generated, we revisit and adjust the coefficients of existing random features. The kernel bandwidth is set as the true bandwidth of the data.

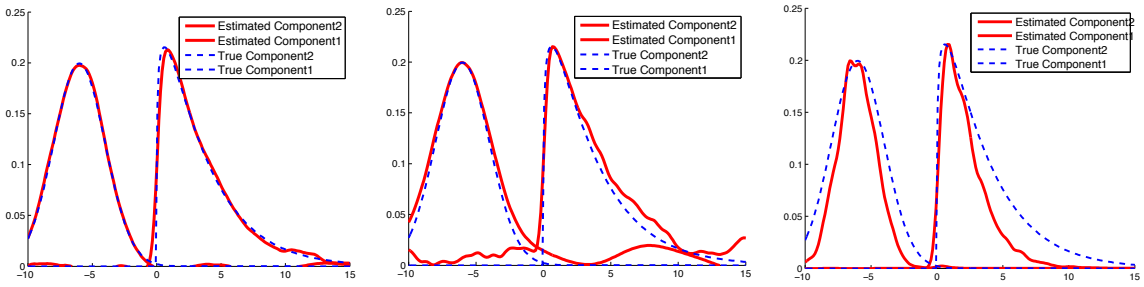
The step size is scheduled as

$$\eta_t = \frac{\theta_0}{1 + \theta_1 t}, \tag{4.27}$$

where θ_0 and θ_1 are two parameters. We use a small $\theta_1 \approx 0.01$ such that in early stages the step size is large enough to arrive at a good initial solution.

Convergence Figure 4.1 shows the convergence rate of the proposed algorithm seeking top $k = 3$ subspace. The potential function is calculated as the squared sine function of the subspace angle between the current solution and the ground-truth. We can see the algorithm indeed converges at the rate $O(1/t)$.

Eigenfunction Recovery Figure 4.2 demonstrate the recovered top $k = 3$ eigenfunctions compared with the ground-truth. We can see the found solution coincides with one eigenfunction, and only disagree slightly on two others.



(a) DSGD-KSVD (b) 2048 random features (c) 2048 Nystrom features

Figure 4.3: Recovered latent components.

4.7.2 Nonparametric Latent Variable Model

In [78], the authors proposed a multiview nonparametric latent variable model that is solved by kernel SVD followed by tensor power iterations. The algorithm can separate latent variables without imposing specific parametric assumptions of the conditional probabilities. However, the scalability of the algorithm was limited by kernel SVD.

Here, we demonstrate that with DSGD-KSVD, we can learn latent variable models with one million data, achieving higher quality of learned components compared with two other approaches.

DSGD-KSVD uses a total of 8192 random features, and in each iteration, it uses a feature mini-batch of size 256 and a data mini-batch of size 512.

We compare with 1) random Fourier features with fixed 2048 functions, and 2) random Nystrom features with fixed 2048 functions. The Nystrom features are calculated by first uniformly sampling 2048 data points, and then evaluate kernel function values on these data points [54].

The dataset consists of two latent components, one is a Gaussian distribution and the other follows a Gamma distribution with shape parameter $\alpha = 1.2$. One million data point are generated from this mixture distribution.

Figures 4.3 shows the learned conditional distributions for each component. We can see DSGD-KSVD achieves almost perfect recovery, while Fourier and Nystrom random

Table 4.3: KCCA results on MNIST 8M (top 50 largest correlations)

# of feat	Random features		Nystrom features	
	corrs.	minutes	corrs.	minutes
256	25.2	3.2	30.4	3.0
512	30.7	7.0	35.3	5.1
1024	35.3	13.9	38.0	10.1
2048	38.8	54.3	41.1	27.0
4096	41.5	186.7	42.7	71.0

DSGD-KCCA		linear CCA	
corrs.	minutes	corrs.	minutes
43.5	183.2	27.4	1.1

feature methods either confuse high density areas or incorrectly estimate the spread of conditional distributions.

4.7.3 KCCA MNIST8M

We then demonstrate the scalability and effectiveness of our algorithm on a large-scale real world dataset. MNIST8M consists of 8.1 million hand-written digits and their transformations. Each digit is of size 28×28. We divide each image into the left and right parts, and learn their correlations using KCCA. Thus the input feature dimension is 392.

The evaluation criteria is the total correlations on the top $k = 50$ canonical correlation directions calculated on a separate test set of size 10000. Out of the 8.1 million training data, we randomly choose 10000 as an evaluation set.

We compare with 1) random Fourier and 2) random Nystrom features on both total correlation and running time. We vary the number of random features used for both methods. Our algorithm uses a total of 20480 features. In each iteration, we use feature mini-batches of size 2048 and data mini-batches of size 1024, and we run 3000 iterations. The kernel bandwidth is set using the "median" trick and is the same for all methods. Due to randomness, all algorithms are run 5 times, and the mean is reported.

The results are presented in Table 4.3. We can see Nystrom features generally achieve better results than Fourier features. Note that for Fourier features, we are using the version

with $\sin$ and $\cos$ pairs, so the real number of parameters is twice the number in the table, as a result the computational time is almost twice of that for Nystrom features.

Our algorithm achieves the best test-set correlations in comparable run time with random Fourier features. This is especially significant for random Fourier features, since the run time would increase by almost four times if double the number of features were used. We can also see that for large datasets, it is important to use more random features for better performance. Actually, the number of random features required should grow linearly with the number of data points. Therefore, our algorithm provides a good balance between the number of random features used and the number of data points processed.

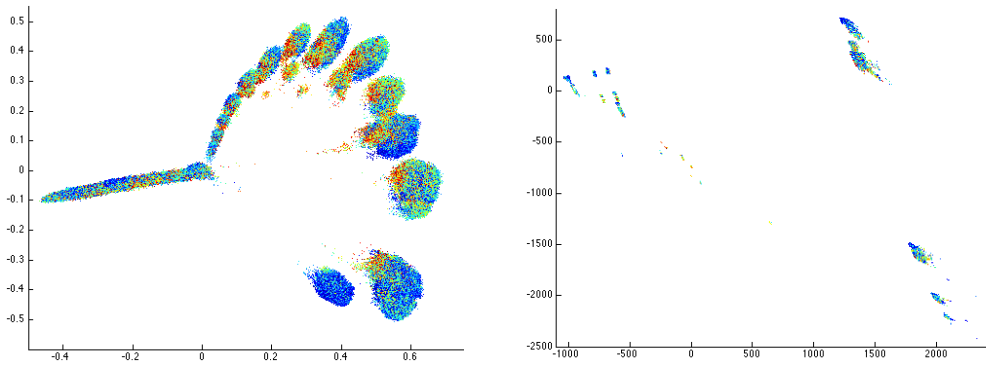
4.7.4 Kernel PCA visualization on molecular space dataset

MolecularSpace dataset contains 2.3 million molecular motifs [60]. We are interested in visualizing the dataset with KPCA. The data are represented by sorted Coulomb matrices of size 75×75 [82]. Each molecule also has an attribute called power conversion efficiency (PCE). We use a Gaussian kernel with bandwidth chosen by the "median trick". We ran kernel PCA with a total of 16384 random features, with a feature mini-batch size of 512, and data mini-batch size of 1024. We ran 4000 iterations with step size $\eta_t = 1/(1 + 0.001 * t)$.

Figure 4.4 presents visualization by projecting the data onto the top two principle components. Compared with linear PCA, KPCA shrinks the distances between the clusters and brings out the important structures in the dataset. We can also see although the PCE values do not necessarily correspond to the clusters, higher PCE values tend to lie towards the center of the ring structure.

4.7.5 Kernel sliced inverse regression on KUKA dataset

We evaluate our algorithm under the setting of kernel sliced inverse regression [80], a way to perform sufficient dimension reduction (SDR) for high dimension regression. After per-



(a) (b)

Figure 4.4: Visualization of the molecular space dataset by the first two principal components. The color corresponds to the PCE value: bluer dots represent lower PCE values while redder dots are for higher PCE values. (a) Kernel PCA; (b) linear PCA. (Best viewed in color)

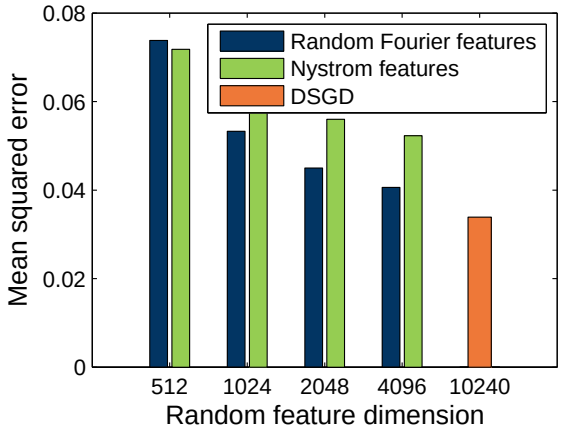


Figure 4.5: Comparison on KUKA dataset.

forming SDR, we fit a linear regression model using the projected input data, and evaluate mean squared error (MSE). The dataset records rhythmic motions of a KUKA arm at various speeds, representing realistic settings for robots [83]. We use a variant that contains 2 million data points generated by the SL simulator. The KUKA robot has 7 joints, and the high dimension regression problem is to predict the torques from positions, velocities and accelerations of the joints. The input has 21 dimensions while the output is 7 dimensions. Since there are seven independent joints, we set the reduced dimension to be seven. We randomly select 20% as test set and out of the remaining training set, we randomly choose 5000 as validation set to select step sizes. The total number of random features is 10240, with mini-feature batch and mini-data batch both equal to 1024. We run a total of 2000 iterations using step size $\eta_t = 15/(1 + 0.001 * t)$.

Figure 4.5 shows the regression errors for different methods. The error decreases with more random features, and our algorithm achieves lowest MSE by using 10240 random features. Nystrom features do not perform as well in this setting probably because the spectrum decreases slowly (there are seven independent joints) as Nystrom features are known to work well for fast decreasing spectrum.

4.8 Summary

We have proposed a general and scalable approach to solve nonlinear component analysis based on doubly stochastic gradients. It is simple, efficient and scalable. In addition, we have theoretical guarantees that the whole subspace converges at the rate $\tilde{O}(1/t)$ to the true subspace. Moreover, since its core is an algorithm for eigenvalue problems in the functional space, it can be applied to various other tasks and models. Finally, we demonstrate the scalability and effectiveness of our algorithm on both synthetic and real world datasets.

CHAPTER 5

COMMUNICATION EFFICIENT DISTRIBUTED KERNEL PCA

In this chapter, we develop a communication efficient algorithm to perform kernel PCA in the distributed setting. The algorithm is a clever combination of subspace embedding and adaptive sampling techniques, and we show that the algorithm can take as input an arbitrary configuration of distributed datasets, and compute a set of global kernel principal components with relative error guarantees independent of the dimension of the feature space or the total number of data points. In particular, computing k principal components with relative error ϵ over s workers has communication cost $\tilde{O}(s\rho k/\epsilon + sk^2/\epsilon^3)$ words, where ρ is the average number of nonzero entries in each data point. Furthermore, we experimented the algorithm with large-scale real world datasets and showed that the algorithm produces a high quality kernel PCA solution while using significantly less communication than alternative approaches.

5.1 Introduction

The original kernel PCA algorithm is designed for a batch setting, where all data points need to fit into a single machine. However, nowadays large volumes of data are being collected increasingly in a distributed fashion, which poses new challenges for running kernel PCA. For instance, a large network of distributed sensors can collect temperature readings from geographically distant locations; a system of distributed data centers in an Internet company can process user queries from different countries; a fraud detection system in a bank needs to perform credit checks on people opening accounts from different branches; and a network of electronic healthcare systems can store patient records from different hospitals. It is very costly in terms of network bandwidth and transmission delays to communicate all of the data collected in a distributed fashion to a single data center, and then

run kernel PCA on the central node. In other words, communication now becomes the bottleneck to the nonlinear feature extraction pipeline. How can we leverage the aggregated computing power in a large distributed system? Can we perform kernel PCA on the entire dataset in a distributed and communication efficient fashion while maintaining provable and strong guarantees in solution quality?

While recent work shows how to do linear PCA in a communication efficient and distributed fashion [84], the kernel setting is significantly more challenging. The main problem with previous work is that it achieves communication proportional to the dimension of the data points, which if implemented straightforwardly in the kernel setting would give communication proportional to the dimension of the feature space which can be very large or even infinite. Kernel PCA uses the kernel trick to avoid going to the potentially infinite dimensional kernel feature space explicitly, so intermediate results are often represented by a function (*e.g.*, a weighted combination) of the feature mapping of some data points. Communicating such intermediate results requires communicating all the data points they depend on. To lower the communication, the intermediate results should only depend on a small number of data points. A distributed algorithm then needs to be carefully designed to meet this constraint.

In this chapter, we propose a communication efficient algorithm for distributed KPCA in a master-worker setting where the dataset is arbitrarily partitioned and each portion sits in one worker, and the workers can communicate only through the master. Our key idea is to design a communication efficient way of generating a small representative subset of the data, and then performing kernel PCA based on this subset. We show that the algorithm can compute a rank-k subspace in the kernel feature space using just a representative subset of size $O(k/\epsilon)$ built in a distributed fashion. For polynomial kernels, it achieves a $(1+\epsilon)$ relative-error approximation to the best rank-k subspace, and for shift-invariant kernels (such as the Gaussian kernel), it achieves $(1+\epsilon)$-approximation with an additive error term that can be made arbitrarily small. In both cases, the total communication for a system of

s workers is $\tilde{O}(s\rho k/\epsilon + sk^2/\epsilon^3)$ words, where ρ is the average number of nonzero entries in each data point, and is always bounded by the dimension of the data d and independent of the dimension of the kernel feature space. This for constant ϵ nearly matches the lower bound $\Omega(sdk)$ for linear PCA [84]. As far as we know, this is the first algorithm that can achieve provable approximation with such communication bounds.

As a subroutine of our algorithm, we have also developed an algorithm for the distributed Column Subset Selection (CSS) problem, which can select a set of $O(k/\epsilon)$ points whose span contains $(1+\epsilon)$-approximation, with communication $O(s\rho k/\epsilon + sk^2)$. This is the first algorithm that addresses the problem for kernels, and it nearly matches the communication lower bound $\Omega(s\rho k/\epsilon)$ for this problem in the linear case [85]. The column subset selection problem has various applications in big data scenarios, so this result could be of independent interest.

Furthermore, our algorithm also leads to some other distributed kernel algorithms: the data can then be projected onto the subspace found and processed by downstream applications. For example, an immediate application is for distributed spectral clustering, that first computes KPCA to rank-k/ϵ and then does k-means on the data projected on the subspace found by KPCA (*e.g.*, [86]). This can be done by combining our algorithm with any efficient distributed k-means algorithms (*e.g.*, [87]).

We evaluate our algorithm on datasets with millions of data points and hundreds of thousands of dimensions where non-distributed algorithms such as batch KPCA are impractical to run. Furthermore, comparing to other distributed algorithms, our algorithm requires less communication and fewer representation data points to achieve the same approximation error.

5.2 Related Work

There has been a surge of recent work on distributed machine learning, *e.g.*, [88, 89, 90, 87]. In this setting, the data sets are typically large, and small error rate is required. This

is because if only a coarse error is needed then there is no need to use large-scale data sets; a small subset of the data will be sufficient. Furthermore, one prefers relative error rates instead of additive error rates, since the latter is worse and harder to interpret without knowing the optimum. Our algorithm can achieve small relative error with limited communication.

Since there exist communication efficient distributed linear PCA algorithms [87, 90], it is tempting to adopt the random feature approach for distributed kernel PCA: first construct m random features and then solve PCA in the primal form, *i.e.*, apply distributed linear PCA on the random features. However, the communication of this method is too high. One needs $m = \tilde{O}(d/\epsilon^2)$ random features to preserve the kernel values up to additive error ϵ, leading to a communication of $O(skm/\epsilon) = O(skd/\epsilon^3)$. Another drawback of using random features is that it only produces a solution in the space spanned by the random features, but not a solution in the feature space of the kernel.

The Nyström method is another popular tool for large-scale kernel methods: sample a subset of data points uniformly at random, and use them to construct an approximation of the original kernel matrix. However, it also suffers from high communication cost, since one needs $O(1/\epsilon^4)$ sampled points to achieve additive ϵ error in the Frobenius norm of the kernel matrix [91]. A closely related method is incomplete Cholesky decomposition [92], where a few pivots are greedily chosen to approximate the kernel matrix. It is unclear how to design a communication efficient distributed version since it requires as many rounds of communication as the number of pivots, which is costly.

Leverage score sampling is a related technique for low-rank approximation [93]. A prior work of Boutsidis et al. [84] gives the first distributed protocol for column subset selection. [94] gives a distributed PCA algorithm with optimal communication cost, but only for linear PCA. In comparison, our work is the first communication efficient distributed algorithm for low rank approximation in the kernel space.

5.3 Backgrounds

For any vector v, let $\|v\|$ denote its Euclidean norm. For any matrix $M \in \mathbb{R}^{d\times n}$, let $M_{i:}$ denote its i-th row and $M_{:j}$ its j-th column. Let $\|M\|_F$ denote its Frobenius norm, and $\|M\|_2$ denote its spectral norm. Let its rank be $r \leq \min\{n, d\}$, and denote its SVD as $M = U\Sigma V^\top$ where $U \in \mathbb{R}^{d\times r}, \Sigma \in \mathbb{R}^{r\times r}$, and $V \in \mathbb{R}^{n\times r}$. Let $[M]_k$ denote its best rank-k approximation. Finally, denote its number of non-zero entries as $\mathrm{nnz}(M)$.

In the distributed setting, there are s workers that are connected to a master processor. Worker i has a local data set $A^i \in \mathbb{R}^{d\times n_i}$, and the global data set $A \in \mathbb{R}^{d\times n}$ is the concatenation of the local data ($n = \sum_{i=1}^s n_i$).

Kernels and Random Features. For a kernel $\kappa(x, x')$, let $\mathcal{H}$ denote its feature space, i.e., there exists a feature mapping $\phi(\cdot) \in \mathcal{H}$ such that $\kappa(x, x') = \langle\phi(x), \phi(x')\rangle_{\mathcal{H}}$. Let $\phi(A) \in \mathcal{H}^n$ denote the matrix obtained by applying ϕ on each column of A and concatenating the results. Throughout this chapter, we regard any $M \in \mathcal{H}^n$ as a matrix whose columns are elements in $\mathcal{H}$ and define matrix operations accordingly. For example, for any $M \in \mathcal{H}^n$ and $N \in \mathcal{H}^m$, let $B = M^\top N \in \mathbb{R}^{n\times m}$ where $B_{ij} = \langle M_{:i}, N_{:j}\rangle_{\mathcal{H}}$, and let $\|M\|_{\mathcal{H}}^2 = \mathrm{tr}\left(M^\top M\right)$. When there is no ambiguity, we omit the subscript $\mathcal{H}$. The random feature approach is a recent technique to scale up kernel methods. Many kernels can be approximated by $\frac{1}{m}\sum_{i=1}^m \xi_{\omega_i}(x)\xi_{\omega_i}(y)$ where ω_i's are randomly sampled. These include Gaussian RBF kernels and other shift-invariant kernels, inner product kernels, etc ([61, 60]).

In this chapter, we provide guarantees for shift-invariant kernels using Fourier random features (the extension to other kernels/random features is straightforward). We assume the kernel satisfies some regularization conditions: it is defined over bounded compact domain in $\mathbb{R}^d$, with $\kappa(0) \leq 1$ and bounded $\nabla^2 k(0)$ [61]. Such conditions are standard in practice, and thus we assume them throughout this chapter.

Kernel PCA. An element $u \in \mathcal{H}$ is an eigenfunction of $\phi(A)\phi(A)^\top$ with the corre-

sponding eigenvalue λ if $\|u\| = 1$ and $\phi(A)\phi(A)^\top u = \lambda u$. Given eigenfunctions $\{u_i\}$ of $\phi(A)\phi(A)^\top$ and eigenvectors $\{v_i\}$ of $\phi(A)^\top\phi(A)$, $\phi(A)$ has the singular decomposition $U\Sigma_k V^\top + U_\perp \Sigma_\perp V_\perp^\top$, where U, V are the lists of top k eigenfunctions/vectors, Σ_k is a diagonal matrix with the corresponding singular values, $U_\perp$, $V_\perp$ are the lists of the rest of the eigenfunctions/vectors, and $\Sigma_\perp$ is a diagonal matrix with the rest of the singular values. Kernel PCA aims to identify the top k subspace U, since the best rank-k approximation $[\phi(A)]_k = U\Sigma_k V^\top = UU^\top\phi(A)$. Typically, the goal is to find a good approximation to this subspace. Formally,

Definition 10 *A subspace $L \in \mathcal{H}^k$ is a rank-k $(1+\epsilon, \Delta)$-approximation for kernel PCA on A if $L^\top L = I_k$ and*

$$\left\|\phi(A) - LL^\top\phi(A)\right\|^2 \leq (1+\epsilon)\left\|\phi(A) - [\phi(A)]_k\right\|^2 + \Delta.$$

Kernel PCA leads to immediate solutions for some other nonlinear component analysis (e.g., kernel CCA), and provides needed subroutines for tasks like spectral clustering.

Subspace Embeddings. Subspace embeddings are a useful technique that can improve the computational and space costs by embedding data into lower dimension while preserving interesting properties. They have been extensively studied in recent years [95, 96, 97]. The recent fast sparse subspace embeddings [97] and its optimizations [98, 99] are particularly suitable for large-scale sparse datasets, since their running time is linear in the number of non-zero entries in the data matrix. They also preserve the sparsity of the input data. Formally,

Definition 11 *An ϵ-subspace embedding of $M \in \mathbb{R}^{m\times n}$ is a matrix $S \in \mathbb{R}^{t\times m}$ such that for any x,*

$$\|SMx\| = (1\pm\epsilon)\|Mx\|.$$

Subspace embeddings can also be done on the right hand side, i.e.*, $S \in \mathbb{R}^{n\times t}$ and $\left\|x^\top MS\right\| = (1\pm\epsilon)\left\|x^\top M\right\|$.*

Mx is in the column space of M and SMx is its embedding, so the definition means that the norm of any vector in the column space of M is approximately preserved. This then provides a way to do dimensional reduction for problems depending on inner products of vectors. Our algorithm repeatedly makes use of subspace embeddings. In particular, the embedding we use is the concatenation of the following known sketching matrices: COUNTSKETCH and i.i.d. Gaussians (or the concatenation of COUNTSKETCH, fast Hadamard and i.i.d. Gaussians). The details can be found in [93]; we only need the following fact.

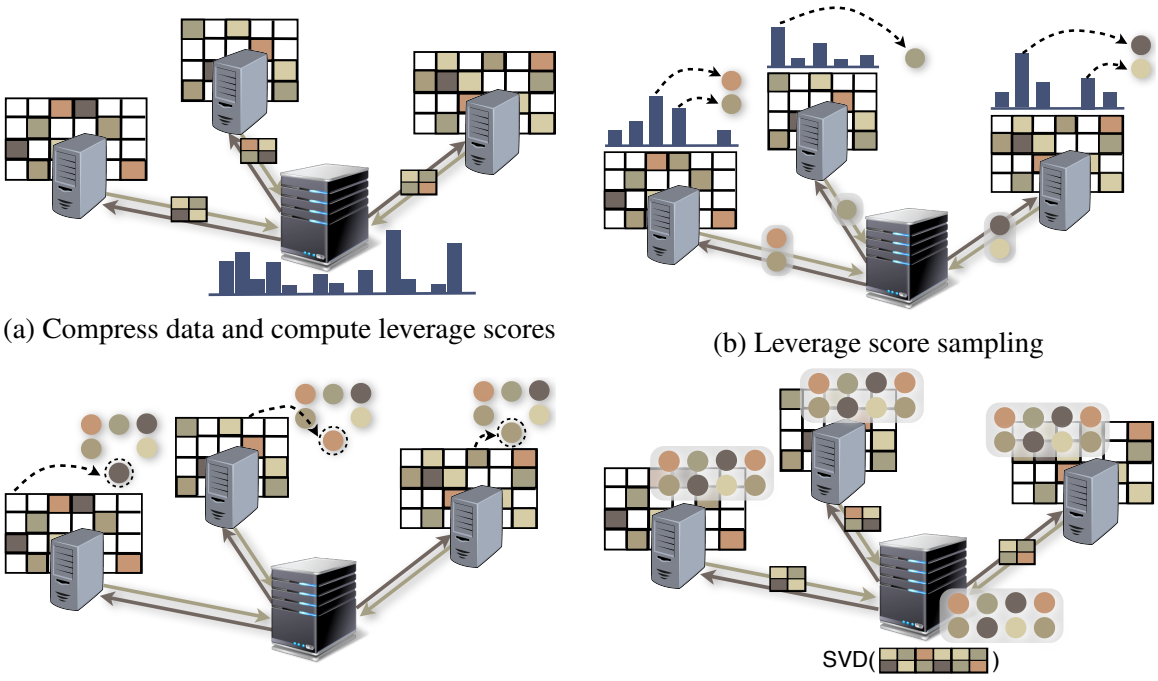
Lemma 12 *For $M \in \mathbb{R}^{d\times n}$, there exist sketching matrices $S \in \mathbb{R}^{t\times d}$ with $t = O(n/\epsilon^2)$ that are ϵ-subspace embeddings. Furthermore, SM can be successfully computed in time $\tilde{O}(\mathrm{nnz}(M))$ with probability at least $1-\delta$.*

The work of [100] shows that a fast computational approach, TENSORSKETCH, is indeed a subspace embedding for the polynomial kernel. However, there are no previously known subspace embeddings for other kernels. We develop efficient and provable embeddings for a large family of kernels including Gaussian kernel and other shift invariant kernels. These embeddings will be a key tool used by our algorithm.

5.4 Overview

In view of the limitations of the related work, we instead take a different approach, which first selects a small subset of points whose span contains an approximation with relative error rate ϵ, and then find a low rank approximation in their span. It is important to keep the size of the subset small and also guarantee that their span contains a good approximation (this is also called kernel column subset selection). A well known technique is to sample according to the statistical leverage scores.

Challenges. However, this immediately raises the following technical challenges.



(a) Compress data and compute leverage scores

(b) Leverage score sampling

(c) Adaptive sampling

(d) Project data and compute KPCA

Figure 5.1: Algorithm overview. The black machine at the center is the master and the gray machines are the workers. Each worker stores its portion of the dataset, and the algorithm computes the top k principle components on the whole dataset. The arrows between the machines denote the direction of communications. In each round, the communication always starts from the workers to the master (lighter arrows) and then from the master to the workers (darker arrows). (a) Each worker compresses its data by using (kernel) subspace embeddings and sends it to the master. The master aggregates the data and computes intermediate results for leverage scores and sends back to the workers. (b) Each worker computes the leverage scores, samples data points (denoted by circles) and then sends them to the master. The master distributes back the union of the sampled data points. (c) Each worker conducts adaptive sampling and sends newly sampled points to the master. The master distributes back the union of all sampled points. (d) Each worker projects its data onto the subspace spanned by the sampled data points and sends the compressed projections to the master. The master computes coefficients for the top k principle components by running SVD, and then sends them back to the workers. (best viewed in color)

I. Computing the statistical leverage scores is prohibitively expensive. Naïvely computing them requires communicating all data points. There exist non-trivial fast algorithms [101], but they are designed for the non-distributed setting. Using them in the distributed setting leads to communication linear in the number of data points, or linear in the number of random features if one uses random features and computes the leverage scores for them.

Our key idea is that it is sufficient to compute the (generalized) leverage scores of the data points, i.e., the leverage scores of another matrix whose row space approximates that of the original data matrix. So the problem is reduced to designing kernel subspace embeddings that can approximate the row space of the data.

II. Even given the embedded data, it is unclear how to compute its leverage scores in a communication efficient way. Although the dimension of the embedded data is small, existing algorithms will lead to communication linear in the number of data points, which is impractical.

III. Simply sampling according to the generalized leverage scores does not give the desired results: a good approximation can only be obtained using a much larger rank, specifically, $O(k/\epsilon)$.

IV. After selecting the small subset of points, we need to design a distributed algorithm to compute a good low rank approximation in their span.

Algorithm. We have designed a distributed kernel PCA algorithm that computes an approximated solution with relative error rate ϵ using low communication. The algorithm operates in following key steps, each of which addresses one of the challenges mentioned above (See Figure 5.1):

I. Kernel Subspace Embeddings. To approximate the subspace of the original data matrix, we propose subspace embeddings for a large family of kernels. For polynomial kernels we improve the prior work by reducing the embedding dimension and thus lowering the communication. Furthermore, we propose new subspace embeddings for kernels

with random feature expansions, allowing PCA for these kernels to be computed in a communication efficient manner. See Section 5.5.1 for the details.

II. Distributed Leverage Scores. To compute the leverage scores, sampling with constant approximations is sufficient. We can thus drastically reduce the number of data points: first do another (non-kernel) subspace embeddings on the embedded data, and then send the result to the master for computing the scores. See Figure 5.1(a) for an illustration and Section 5.5.2 for the details.

III. Sampling Representative Points. We take a two-step approach as leverage scores alone is not good enough : first sample according to generalized leverage scores, and then sample additional points according to their distances to the span of the points sampled in the first step. The first step gains some coarse information about the data, and the second step use it to get the desired samples. The two steps are illustrated in Figure 5.1(b) and 5.1(c), respectively, while the details are in Section 5.5.3.

IV. Computing an Approximation. After projecting the data to the span of the representative points, we sketch the projections by (non-kernel) subspace embeddings. We then send the compressed projections to the master and compute the solution there. See Figure 5.1(d) for an illustration and Section 5.5.4 for the details.

Main Theoretical Results. Given as input the local datasets, the rank k and error parameters ϵ, Δ, our algorithm outputs a $(1+\epsilon, \Delta)$-approximation to the optimum with large probability. Formally,

Theorem 13 *Algorithm 7 produces a subspace L for kernel PCA on A that with probability ≥ 0.99 satisfies:*

1. *L is a rank-k $(1+\epsilon, 0)$-approximation when applied to polynomial kernels.*
2. *L is a rank-k $(1+\epsilon, \Delta)$-approximation when applied to shift-invariant kernels with regularization.*

The total communication is $\tilde{O}(\frac{s\rho k}{\epsilon} + \frac{sk^2}{\epsilon^3})$ words, where ρ is the average number of nonzero entries in one data point.

The constant success probability can be boosted up to any high probability $1 - \delta$ by repetition, which adds only an extra $O(\log \frac{1}{\delta})$ term to communication and computation.

The output subspace L is represented by $\tilde{O}(k/\epsilon)$ sampled points Y from A (*i.e.*, $L = \phi(Y)C$ for some coefficient matrix C), so L can be easily communicated and the projection of any point on L can be easily computed by the kernel trick. The communication has linear dependence on the dimension and the number of workers, and has no dependence on the number of data points, which is crucial for big data scenarios. Moreover, it does not depend on Δ (but the computation does), so the additive error can be made arbitrarily small with more computation.

The theorem also holds for other properly regularized kernels with random feature expansions (see [61, 60] for more such kernels); the extension of our proof is straightforward.

We also make the following contributions: (*i*) Subspace embedding techniques for many kernels. (*ii*) Distributed algorithm for computing generalized leverage scores with low communication. (*iii*) Distributed algorithm for kernel column subset selection.

5.5 Distributed Kernel Principal Component Analysis

Our algorithm first computes the (generalized) leverage scores that measure the non-uniform structure, then samples the desired subset of points whose span contains a good approximated solution, and finally finds such a solution in the span.

Leverage scores are critical for importance sampling in many fast randomized algorithms. The leverage scores are defined as follows.

Definition 14 *For $E \in \mathbb{R}^{t\times n}$ with SVD $E = U\Sigma V^\top$, the leverage score ℓ_j for its j-th column is $\ell_j = \|V_{j:}\|^2$.*

Their importance is reflected in the following fact: suppose E has rank at most k, and suppose P is a subset of $O(\frac{k\log k}{\epsilon^2})$ columns obtained by repeatedly sampled from the columns of E according to their leverage scores, then the span of P contains an $(1+\epsilon, 0)$-approximation subspace for E with probability ≥ 0.99 (see,*e.g.*, [102]). Here, sampling

one column according to the leverage scores ℓ_j means to define sampling probabilities p_j such that $p_j \geq \frac{\ell_j}{4\sum_j \ell_j}$ for all j, and then pick one column where the j-th column is picked with probability p_j. Note that setting $p_j = \frac{\ell_j}{\sum_j \ell_j}$ is clearly sufficient, but a constant variance of p_j is allowed at the expense of an extra constant factor in the sample size. This means that it is sufficient to compute constant approximations $\tilde{\ell}_j$ for ℓ_j, and then sample according to $p_j = \frac{\tilde{\ell}_j}{\sum_j \tilde{\ell}_j}$.

However, even computing constant approximations of the leverage scores are non-trivial: naïve approaches require SVD, which is expensive. Actually, SVD is more expensive than the task of PCA itself. Even ignoring computation cost, naïve SVD is prohibitive in the distributed setting due to its high communication cost. Fortunately, it turns out that the leverage scores are an over kill for our purpose; it suffices to compute the generalized leverage scores, i.e., the leverage scores of a proxy matrix.

Definition 15 *If E has rank q and can approximate the row space of M up to $(1+\epsilon, \Delta)$,* i.e.*, there exists X with*

$$\|XE - M\|_F \leq (1+\epsilon)\|M - [M]_k\|_F + \Delta,$$

then the leverage scores of E are called the generalized leverage scores of M with respect to rank q.

This generalizes the definition in [101] by allowing the rank of E to be larger than k and allowing additive error Δ, which are important for our application. The generalized leverage scores can act as the leverage scores for our purpose in the following sense.

Lemma 16 *Let P be $O(\frac{q \log q}{\epsilon^2})$ columns sampled from M according to their generalized leverage scores w.r.t. rank q. Then with probability ≥ 0.99, the span of P has a rank-s $(1+2\epsilon, 2\Delta)$-approximation subspace for M.*

Proof It follows from combining Theorem 5 in [102] and the definition of the generalized leverage scores. ■

Computing the generalized scores with respect to rank q could be much more efficient, since the intrinsic dimension now becomes q, which can be much smaller than the ambient dimension (the number of points or the dimension of the feature space). However, as noted in the overview, there are still a few technical challenges.

- Efficiently find a smaller matrix E that can approximate the row space of the original data.
- Compute the leverage scores of E in a communication efficient way.
- The approximation solution in the span of P has the same rank as E, which is $O(k/\epsilon)$ when we use kernel subspace embedding to obtain E. This is not satisfying since our final goal is to compute a rank-k solution.
- Find a good approximation in the span of $\phi(Y)$ with low communication.

Our final algorithm consists of four key steps, each of which addresses one of the above challenges. They are elaborated in the following four subsections respectively, and the final subsection presents the overall algorithm.

5.5.1 Kernel Subspace Embeddings

Recall that a subspace embedding S for a matrix M is such that $\|SMx\| \approx \|Mx\|$, *i.e.*, the norm of any vector in the column space of M is approximately preserved. Subspace embeddings can also be generalized for the feature mapping of kernels, simply by setting $M = \phi(A)$, S a linear mapping from $\mathcal{H} \mapsto \mathbb{R}^t$ and using the corresponding inner product. If the data after the kernel subspace embedding is sufficient for solving the problem under consideration, then only $S\phi(A)$ in much lower dimension is needed. This is especially interesting for distributed kernel methods, since directly using the feature mapping or the kernel trick in this setting will lead to high communication cost, while the data after embedding can be much smaller and lead to much lower communication cost.

A sufficient condition for solving many problems (in particular, kernel PCA) is to preserve the low rank structure of the data. More precisely, the row space of $S\phi(A)$ is a good approximation to that of $\phi(A)$, where the error is comparably to the best rank k approximation error. Then $S\phi(A)$ can be used to compute the generalized leverage scores for $\phi(A)$, which can then be utilized to compute kernel PCA as mentioned above.

More precisely, we would like $S\phi(A)$ to approximate the row space of $\phi(A)$ up to $(1+\epsilon, \Delta)$, as required in the definition of the generalized leverage scores. We give such embeddings a particular name.

Definition 17 *S is called a $(1+\epsilon, \Delta)$-good subspace embedding for $\phi(A) \in \mathcal{H}^n$, if there exists X such that*

$$\|X(S\phi(A)) - \phi(A)\|^2 \leq (1+\epsilon)\|\phi(A) - [\phi(A)]_k\|^2 + \Delta.$$

We now identify the sufficient conditions for $(1+\epsilon, \Delta)$-good subspace embeddings, which can then be used in constructing such embeddings for various kernels.

Lemma 18 *S is a $(1+\epsilon, \Delta)$-good subspace embedding for $\phi(A) \in \mathcal{H}^n$ if it satisfies the following.*

P1 *(Subspace Embedding): For any orthonormal $V \in \mathcal{H}^k$ (*i.e.*, $V^\top V$ is the identity), for all $x \in \mathbb{R}^k$,*

$$\|SVx\| = (1 \pm c)\|Vx\|$$

where c is a sufficiently small constant.

P2 *(Approximate Product): for any $M \in \mathcal{H}^n, N \in \mathcal{H}^k$,*

$$\left\|(SN)^\top(SM) - N^\top M\right\|_F^2 \leq \frac{\epsilon}{k}\|N\|^2\|M\|^2 + \Delta.$$

Polynomial Kernels. For polynomial kernels, there exists an efficient algorithm, namely TENSORSKETCH to compute the embedding [100]. However, the embedding dimension

has a quadratic dependence on the rank k, which will increase the communication. Fortunately, subspace embedding can be concatenated, so we can further apply another known subspace embedding such as one of those in Lemma 12 which, though not fast for feature mapping, is fast for the already embedded data and has lower dimension. In this way, we can enjoy the benefits of both approaches.

The guarantee of TENSORSKETCH in [100] and the property of the subspace embeddings in Lemma 12 can be combined to verify **P1** and **P2**. So we have

Lemma 19 *For polynomial kernels $\kappa(x, y) = (\langle x, y\rangle)^q$, there exists a $(1+\epsilon, 0)$-good subspace embedding matrix $S : \mathbb{R}^{d^q} \mapsto \mathbb{R}^t$ with $t = O(k/\epsilon)$.*

Kernels with Random Feature Expansions. Polynomial kernels have finite dimensional feature mappings, for which the sketching seems natural. It turns out that it is possible to extend subspace embeddings to kernels with infinite dimensional feature mappings. More precisely, we propose subspace embeddings for kernels with random feature expansions, *i.e.*, $\kappa(x, y) = \mathbb{E}_\omega [\xi_\omega(x)\xi_\omega(y)]$ for some function $\xi(\cdot)$. Therefore, one can approximate the kernel by using m features $z_\omega(x)$ on randomly sampled ω. Such random feature expansion can be exploited for subspace embeddings: view the expansion as the "new" data points and apply a sketching matrix on top of it. Compared to polynomial kernels, the finite random feature expansion leads to an additional additive error term. Our analysis shows that bounding the additive error term only requires sufficiently large sampled size m, which affects the computation but does not affect the final embedding dimension and thus the communication.

In summary, the embedding is $S\phi(x) = TR(\phi(x))$, where $R(\phi(x)) \in \mathbb{R}^m$ is m random features for x and $T \in \mathbb{R}^{t\times m}$ is an embedding as in Lemma 12. The properties **P1** and **P2** can be verified by combining Lemma 12 and the guarantees of random features.

Lemma 20 *For a continuous shift-invariant kernels $\kappa(x, y) = \kappa(x-y)$ with regularization, there exists a $(1+\epsilon, \Delta)$-good subspace embedding $S : \mathcal{H} \mapsto \mathbb{R}^t$ with $t = O(k/\epsilon)$.*

Algorithm 4 Distributed Leverage Scores:
$\{\tilde{\ell}_j^i\} = \textbf{disLS}(\{E^i\}_{i=1}^s, k)$

1: Each worker i: do $\frac{1}{4}$-subspace embedding $E^iT^i \in \mathbb{R}^{t\times p}$ with $p = O(t)$; send E^iT^i to Master.
2: Master: QR-factorize $\left[E^1T^1, \ldots, E^sT^s\right]^\top = UZ$;
send Z to all workers.
3: Each worker i: compute $\tilde{\ell}_j^i = \left\|\left((Z^\top)^{-1}E^i\right)_{:j}\right\|_2^2$.

5.5.2 Computing Leverage Scores

Given the matrix E obtained from kernel subspace embedding, we would like to compute the leverage scores of E. First note that this cannot be done simply in a local manner: the leverage score of a column in E^i is different from the leverage score of the same column in E. Furthermore, though data in E have low dimension, communicating all points in E to the master is still impractical, since it leads to communication linear in the total number of points.

Fortunately, we only need to compute constant approximations of the scores, which allows us to use subspace embedding on E to greatly reduce the number of data points. In particular, we apply a $\frac{1}{4}$-subspace embedding T^i (*e.g.*, one of those in Lemma 12) on each local data set E^i, and then send them to the master. Let ET denote all the embedded data, and do QR factorization $(ET)^\top = UZ$. Now, the rows of $U^\top = \left(Z^\top\right)^{-1} ET$ are a set of basis for ET. Then, think of $U^\top T^\dagger = \left(Z^\top\right)^{-1} E$ as the basis for E, so it suffices to compute the norms of the columns in $\left(Z^\top\right)^{-1} E$.

The details are described in Algorithm 4 and Figure 5.1(a) shows an illustration. The algorithm is guaranteed to output constant approximations of the leverage scores of E.

Lemma 21 *Let ℓ_j^i be the true leverage scores of E. Then Algorithm 4 outputs $\tilde{\ell}_j^i = (1 \pm 1/2)\ell_j^i$.*

Proof The algorithm can be viewed as applying an embedding $T = \operatorname{diag}\left(T^1, \ldots, T^s\right)$ on E to approximate the scores while saving the costs. Each T^i is an $\frac{1}{4}$-subspace embedding

Algorithm 5 Sampling Representative Points:
$Y = \textbf{RepSample}(\{A^i\}_{i=1}^s, \{\tilde{\ell}_j^i\}, k, \epsilon)$

1: Workers: sample $O(k \log k)$ points according to $\{\tilde{\ell}_j^i\}$;
send to Master;
2: Master: send all the sampled points P to the workers;
3: Workers: sample $O(k/\epsilon)$ points $\tilde{Y}$ according to the square distances to P in the feature space;
send to Master;
4: Master: send $Y = \tilde{Y} \cup P$ to all the workers.

matrix, then for any x,

$$\begin{aligned}\left\|x^\top ET\right\|^2 &= \left\|[x^\top E^1 T^1, x^\top E^2 T^2, \ldots, x^\top E^s T^s]\right\|^2 \\ &= \sum_{i=1}^s \left\|x^\top E^i T^i\right\|^2 = \sum_{i=1}^s (1 \pm 1/4)^2 \left\|x^\top E^i\right\|^2 \\ &= (1 \pm 1/4)^2 \left\|x^\top E\right\|^2.\end{aligned}$$

So T is also $\frac{1}{4}$-subspace embedding. Such a scheme of using embedding for approximating the scores has been analyzed (Lemma 6 in [101]), and the lemma follows. ■

We note that though a constant approximation is sufficient for our purpose, but the algorithm can output $\tilde{\ell}_j^i = (1 \pm \epsilon)\ell_j^i$ by doing an $\frac{\epsilon}{2}$-subspace embedding (instead of $\frac{1}{4}$), which can be useful for other applications.

5.5.3 Sampling Representative Points

Sampling directly to the leverage scores can produce a set of points P such that the span of $\phi(P)$ contains a $(1+\epsilon, \Delta)$-approximation to $\phi(A)$. However, the rank of that approximation can be as high as $O(k/\epsilon)$, since its rank is the same as that of the embedded data (see Lemma 16), which will be $O(k/\epsilon)$ to achieve ϵ error. To get a rank-k approximation and also enjoy the advantage of leverage scores, we propose to combine leverage score sampling and the adaptive sampling algorithm in [103, 104].

Algorithm 6 Computing an Approximation:
$L = \textbf{disLR}(\{A^i\}_{i=1}^s, Y, k, \epsilon, \Delta)$

1: Each worker i: compute the basis Q for $\phi(Y)$ and $\Pi^i = Q^\top \phi(A^i)$; do an ϵ-subspace embedding $\Pi^i T^i \in \mathbb{R}^{|Y| \times w}$ with $w = O(|Y|/\epsilon^2)$, and send $\Pi^i T^i$ to Master;
2: Master: concatenate $\Pi T = [\Pi^1 T^1, \ldots, \Pi^s T^s]$ and send the top k singular vectors W of ΠT to the workers.
3: Each worker i: set $L = QW$.

The details are presented in Algorithm 5. We first sample a set P of $O(k \log k)$ points according to the leverage scores, so that the span of $\phi(P)$ contains a $(2, \Delta)$-approximation. Then we use the adaptive sampling method: sample $O(k/\epsilon)$ points according to the square distances from the points to their projections on P and then add them to P to get the desire set Y of representative points. Figure 5.1(b) and 5.1(c) demonstrate the two steps of the algorithm.

Adaptive sampling has the following guarantee:

Lemma 22 *Suppose there is a $(2, \Delta)$-approximation for $\phi(A)$ in the span of $\phi(P)$. Then with probability ≥ 0.99, the span of $\phi(Y)$ has a rank-k $(1 + \epsilon, \Delta)$-approximation.*

Therefore, we solves the column subset selection problem for kernels in the distributed setting, with $O(k \log k + k/\epsilon)$ selected columns and with a communication of only $O(s\rho k/\epsilon + sk^2)$. This also provides the foundation for kernel PCA task.

5.5.4 Computing an Approximation

To compute a good approximation in the span of $\phi(Y)$, the naïve approach is to project the data to the span and compute SVD there. However, the communication will be linear in the number of data points. Subspace embedding can be used to sketch the projected data, so that the number of data points is greatly reduced.

Algorithm 7 describes the details and Figure 5.1(d) shows an illustration. To compute the best rank-k approximation for the projected data Π, we do a subspace embedding on the right hand side, *i.e.*, compute $\Pi T = [\Pi^1 T^1, \ldots, \Pi^s T^s]$. Then the algorithm computes

the best rank-k approximation W for ΠT, which is then a good approximation for Π and thus $\phi(A)$. It then returns L, the representation of W in the coordinate system of $\phi(A)$. The output L is guaranteed to be a good approximation.

Lemma 23 *If there is a rank-k $(1+\epsilon, \Delta)$-approximation subspace in the span of $\phi(Y)$, then*

$$\left\|LL^\top\phi(A) - \phi(A)\right\|^2 \leq (1+\epsilon)^2 \left\|\phi(A) - [\phi(A)]_k\right\|^2 + (1+\epsilon)\Delta.$$

Proof Sketch. For our choice of w, T^i is an ϵ-subspace embedding matrix for Π^i. Then their concatenation B is an ϵ-subspace embedding for Π, the concatenation of Π^i. Then we can apply the idea implicit in [90].

By Pythagorean Theorem, the error can be factorized into

$$\underbrace{\left\|LL^\top\phi(A) - QQ^\top\phi(A)\right\|^2}_{T1} + \underbrace{\left\|\phi(A) - QQ^\top\phi(A)\right\|^2}_{T2}.$$

Since $LL^\top = QWW^\top Q^\top$,

$$T1 = \left\|WW^\top Q^\top\phi(A) - Q^\top\phi(A)\right\|^2.$$

Note that $\Pi = Q^\top\phi(A)$, and W is the best rank-k subspace for its embedding ΠT. By property of T (Theorem 7 in [90]), it is also a good approximation for Π. So

$$T1 \approx \left\|[Q^\top\phi(A)]_k - Q^\top\phi(A)\right\|^2 = \left\|Q[Q^\top\phi(A)]_k - QQ^\top\phi(A)\right\|^2.$$

Combining this with $T2$, and applying Pythagorean Theorem again, we know that the error is roughly

$$\left\|Q[Q^\top\phi(A)]_k - \phi(A)\right\|^2.$$

Now, by assumption, there is a rank-k $(1+\epsilon, \Delta)$-approximation subspace X in the span

Algorithm 7 Distributed Kernel PCA:
$L =$ **disKPCA**$(\{A^i\}_{i=1}^s, k, \epsilon, \Delta)$

1: Each worker i: do a $(1/4, \Delta)$-good subspace embedding $E^i = S(\phi(A^i)) \in \mathbb{R}^{t \times n_i}, t = O(k)$;
2: Compute the leverage scores: $\{\tilde{\ell}_j^i\} =$ **disLS**$(\{E^i\}_{i=1}^s, k)$;
3: Sample points: $Y =$ **RepSample**$(\{A^i\}_{i=1}^s, \{\tilde{\ell}_j^i\}, k, \epsilon)$;
4: Output $L =$ **disLR**$(\{A^i\}_{i=1}^s, Y, k, \epsilon, \Delta)$.

of $\phi(Y)$. Since $[Q^\top \phi(A)]_k$ is the best rank-k approximation to $Q^\top \phi(A)$,

$$\begin{aligned}
&\left\|Q[Q^\top \phi(A)]_k - \phi(A)\right\|^2 \\
=& \left\|Q[Q^\top \phi(A)]_k - QQ^\top \phi(A)\right\|^2 + \left\|QQ^\top \phi(A) - \phi(A)\right\|^2 \\
\leq& \left\|X - QQ^\top \phi(A)\right\|^2 + \left\|QQ^\top \phi(A) - \phi(A)\right\|^2 \\
=& \left\|X - \phi(A)\right\|^2.
\end{aligned}$$

The lemma then follows. ■

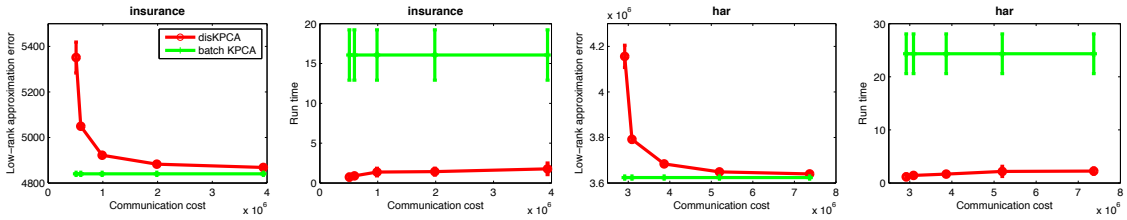
5.5.5 Overall Algorithm

Now, putting things together, we obtain our final algorithm for distributed kernel PCA (Algorithm 7). Our main result, Theorem 13, follows by combining all the lemmas in the previous subsections (with properly adjusted ϵ and Δ).

5.6 Experiments

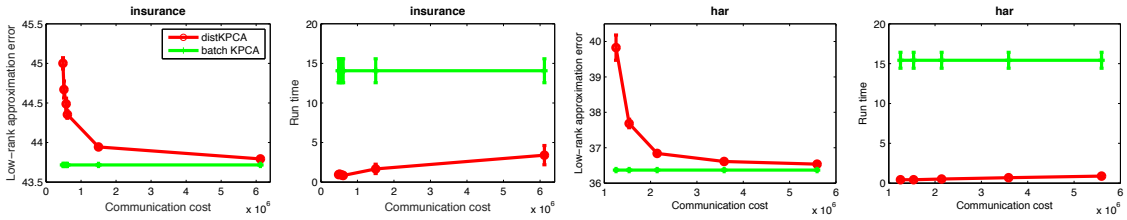
5.6.1 Datasets

We use ten datasets to evaluate our algorithm. They contain both sparse and dense data and come from a variety of different domains, such as text, images, high energy physics and biology. We use two smaller ones to benchmark against the single-machine batch KPCA



(a) error on insurance (b) runtime on insurance (c) error on har (d) runtime on har

Figure 5.2: KPCA for polynomial kernels on small datasets: low-rank approximation error and runtime



(a) error on insurance (b) runtime on insurance (c) error on har (d) runtime on har

Figure 5.3: KPCA for Gaussian kernels on small datasets: low-rank approximation error and runtime

Table 5.1: Dataset specification: d is the original feature dimension, n is the number of data points, and s is the total number of workers storing the dataset distributedly. Among them, bow and 20news are sparse datasets. All datasets except mnist8m are taken from UCI repository [105] and [106].

Dataset	d	n	s
bow	100,000	8,000,000	200
higgs	28	11,000,000	200
mnist8m	784	8,000,000	100
susy	18	5,000,000	100
yearpredmsd	90	463,715	10
ctslice	384	53,500	10
20news	61,118	11,269	5
protein	9	41,157	5
har	561	10,299	5
insurance	85	9,822	5

algorithm while the rest are large-scale datasets with up to tens of millions of data points and hundreds of thousands dimensions. Refer to Table 5.1 for detailed specifications.

Each dataset is partitioned on different workers according to the power law distribution with exponent 2 to simulate the distribution of the data over large networks [107]. Depending on the size of the dataset, the number of workers used ranges from 5 to 200 (see Table 5.1 for details).

5.6.2 Experiment Settings

Since our key contribution is sampling a small set of data points intelligently, the natural alternative is uniformly sampling. We compare with two variants of uniform sampling algorithms: 1) uniformly sampling representative points and use Algorithm 6 to get KPCA solution (denoted as uniform+disLR); 2) uniformly sampling data points and apply batch KPCA (denoted as uniform+batch KPCA).

For both algorithms, we compare the tradeoff of low rank approximation error and communication cost. Particularly, we compare the communication needed to achieve the same error. Each method is run 5 times and the mean and the standard deviation are reported.

For polynomial kernel, the degree is $q = 4$ and for Gaussian RBF kernel, the kernel bandwidth σ is set to 0.2 of the median pairwise distance among a subset of 20000 randomly chosen data points (a.k.a, the "median trick"). For Gaussian random feature expansion, we use 2000 random features.

In all experiments, we set the number of principle components $k = 10$, which is the same number for k-means. The algorithm specific parameters are set as follows: 1) The subspace embedding dimension for the feature expansion t is 50; 2) The subspace embedding dimension for the data points p is 250; 3) We vary the number of adaptively sampled points $|\tilde{Y}|$ from 50 to 400 to simulate different communication cost; 4) The subspace embedding dimension w is set to equal $|Y|$.

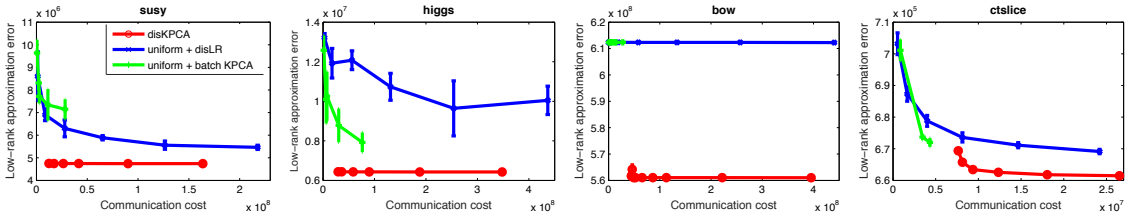


Figure 5.4: KPCA for polynomial kernels on larger datasets

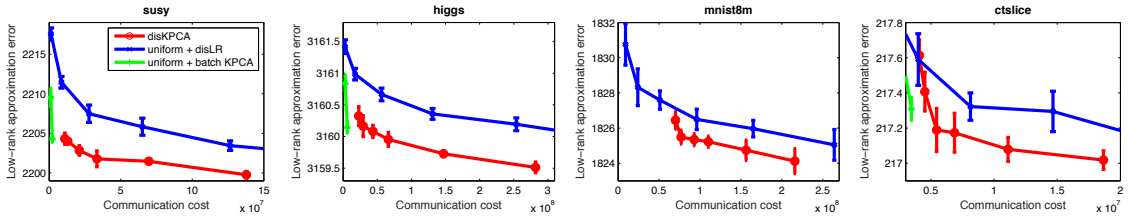


Figure 5.5: KPCA for Gaussian kernels on larger datasets

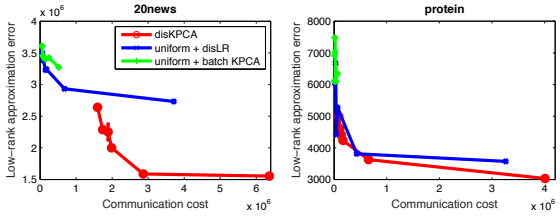


Figure 5.6: KPCA results for arc-cos kernels

5.6.3 Comparison with Batch Algorithm

We compare to the "ground-truth" solutions produced by batch KPCA on two small datasets where it is feasible. The experiment results for the polynomial kernel and the Gaussian RBF kernel are presented in Figures 5.2 and 5.3, respectively. In both cases, the approximation error of disKPCA decreases as more communication is allowed. It can nearly match the optimum low-rank approximation error with much fewer data points. In addition, it is much faster: we gain a speed up of $10\times$ by using five workers.

5.6.4 Communication Efficiency

In these experiments, we compare the tradeoff between communication cost and approximation accuracy on large-scale datasets. The alternative, uniform + batch KPCA, is stopped short in many experiments due to its excessive computation cost for larger number of sam-

pled data points.

Figure 5.4 demonstrates the performance on polynomial kernels on four large datasets. On all four datasets, our algorithm outperforms the alternatives by significant margins. Especially on bow, which is a sparse dataset, the usage of kernel embeddings takes advantage of the sparsity structure and leads to much smaller error. On other datasets, uniform + disLR cannot match the error achieved by our algorithm even when using much more communication.

Figure 5.5 shows the performance on Gaussian kernels. On mnist8m, the error for uniform + batch KPCA is so large (almost twice of the errors in the figure) that it is not shown. On other datasets, disKPCA achieves significant smaller error. For example, on higgs dataset, to achieve the same error, uniform + disLR requires more than 5 times communication. Since it does not have the communication of computing leverage scores, this means that it needs to sample much more points to get similar performance. Therefore, our algorithm is very efficient in communication.

Besides polynomial and Gaussian kernels, we have also conducted experiments using arc-cos kernel [108]. The arc-cosine kernels have random feature bases similar to the Rectified Linear Units (ReLU) used in deep learning. We use degree $n = 2$ and Figure 5.6 shows the results. Our algorithm consistently achieves better tradeoff between communication and approximation and the benefit is especially more pronounced on sparser dataset such as 20news.

5.6.5 Scaling Results

In Figure 5.7, we present the scaling results for disKPCA. In these experiments, we vary the number of workers and record the corresponding computation time (communication time excluded). On both datasets, the runtime decreases as we use more workers, and it eventually plateaus. Our algorithm gains about $2\times$ speedup by using $4\times$ more workers. Note that our algorithm is designed to strike a good balance between communication and approxima-

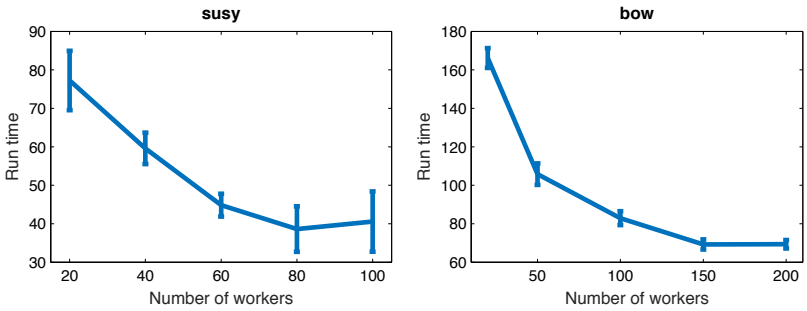


Figure 5.7: KPCA scaling results

tion. Even though computation complexity is not our first priority, the experiments show disKPCA still enjoys favorable scaling property.

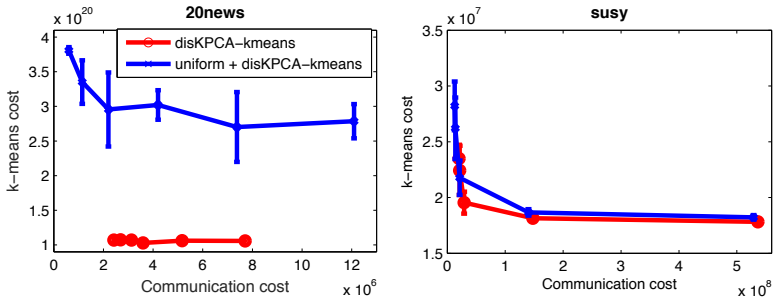
5.6.6 Distributed Spectral Clustering

We have also experimented a form of spectral clustering (KPCA followed by k-means clustering). We project the data onto the top k principle components and then apply a distributed k-means clustering algorithm [87]. The evaluation criterion is the k-means objective, *i.e.*, average distances to the corresponding centers, in the feature space.

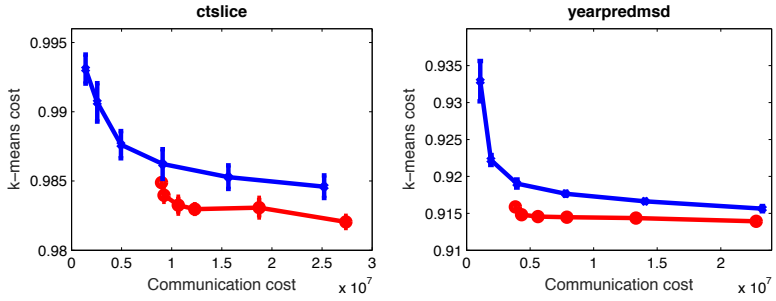
Figure 5.8a presents results for polynomial kernels on the 20news and susy and Figure 5.8b presents results for Gaussian kernels on ctslice and yearpredmsd. Our algorithm compares favorably with the other methods and achieves a better tradeoff of communication and error. This means that although the other methods require similar communication, they need to sample more data points to achieve the same loss, demonstrating the effectiveness of our algorithm.

5.7 Summary

This chapter proposes a communication efficient distributed algorithm for kernel Principal Component Analysis with theoretical guarantees. It computes a relative-error approximation compared to the best rank-k subspace, using communication that nearly matches that of the state-of-the-art algorithms for distributed linear PCA. This is the first distributed al-



(a) Polynomial kernels



(b) Gaussian kernels

Figure 5.8: KPCA + k-means clustering

gorithm that can achieve such provable approximation and communication bounds. The experimental results show that it can achieve better performance than the baseline using the same communication budget.

PART III: ANALYZING NEURAL NETWORKS

In the previous two parts of the thesis, we have discussed how to solve non-convex problems in the context of latent variable models and how to scale up nonlinear eigen decomposition. Another important class of non-convex problems in machine learning is deep neural networks. Neural networks have gained much more attention due to their success in various applications such as computer vision, natural language understanding, and reinforcement learning, *etc.*.

The most common way of learning the neural networks is stochastic gradient descent (SGD), which is a variant of gradient descent. Although it is a highly non-convex problem, SGD seems to perform surprisingly well in practice. Given the seemingly mismatch between theory and practice, it is therefore of great importance to understand in more details why gradient descent works so well for neural networks. In this part, we introduce some novel analysis on this important question.

In Chapter 6, we study one-hidden-layer neural network with rectified linear units (ReLU) and show that with high probability and with diverse weights, the optimization landscape does not have spurious local optima. To approach the problem, we directly analyze the gradient and re-write it as a special matrix times the residual vector. As a result, if we can control the spectrum of the matrix, we have an upper bound on the final error.

Based on these insights, we introduce semi-random units in Chapter 7. This special unit sits between fully adjustable ReLU units and fixed random features. The benefit of semi-random features is that it eliminates the diversity requirement on the neural weights, which is difficult to enforce in gradient descent. At the same, semi-random features achieve much lower approximation error compared with random features and they only require slightly more units to reach comparable performance as fully-adjustable ReLU units.

CHAPTER 6
ONE-HIDDEN-LAYER NEURAL NETWORK HAS NO SPURIOUS LOCAL MINIMA

Neural networks are a powerful class of functions that can be trained with simple gradient descent to achieve state-of-the-art performance on a variety of applications. Despite their practical success, there is a paucity of results that provide theoretical guarantees on why they are so effective. Lying in the center of the problem is the difficulty of analyzing the non-convex loss function with potentially numerous local minima and saddle points. Can neural networks corresponding to the stationary points of the loss function learn the true target function? If yes, what are the key factors contributing to such nice optimization properties?

In this chapter, we answer these questions by analyzing one-hidden-layer neural networks with ReLU activation, and show that despite the non-convexity, neural networks with diverse units have no spurious local minima. We bypass the non-convexity issue by directly analyzing the first order optimality condition, and show that the loss can be made arbitrarily small if the minimum singular value of the "extended feature matrix" is large enough. We make novel use of techniques from kernel methods and geometric discrepancy, and identify a new relation linking the smallest singular value to the spectrum of a kernel function associated with the activation function and to the diversity of the units. Our results also suggest a novel regularization function to promote unit diversity for potentially better generalization.

6.1 Introduction

Neural networks are a powerful class of nonlinear functions which have been successfully deployed in a variety of machine learning tasks. In the simplest form, neural networks with

one hidden layer are linear combinations of nonlinear basis functions (units),

$$f(x) = \sum_{k=1}^{n} v_k \sigma(w_k^\top x) \tag{6.1}$$

where $\sigma(w_k^\top x)$ is a basis function with weights w_k, and v_k is the corresponding combination coefficient. Learning with neural networks involves adapting both the combination coefficients and the basis functions at the same time, usually by minimizing the empirical loss

$$L(f) = \frac{1}{2m} \sum_{l=1}^{m} \ell(y_l, f(x_l)) \tag{6.2}$$

with first-order methods such as (stochastic) gradient descent. It is believed that basis function adaptation is a crucial ingredient for neural networks to achieve more compact models and better performance [109, 110].

However, the empirical loss minimization problem involved in neural network training is non-convex with potentially numerous local minima and saddle points. This makes formal analysis of training neural networks very challenging. Given the empirical success of neural networks, a sequence of important and urgent scientific questions need to be investigated: Can neural networks corresponding to stationary points of the empirical loss learn the true target function? If the answer is yes, then what are the key factors contributing to their nice optimization properties? Based on these understandings, can we design better regularization schemes and learning algorithms to improve the training of neural networks?

In this chapter, we provide partial answers to these questions by analyzing one-hidden-layer neural networks with rectified linear units (ReLU) in a least-squares regression setting. We show that neural networks with diverse units have no spurious local minima. More specifically, we show that the training loss of neural networks decreases in proportion to $\|\partial L/\partial W\|^2 / s_m^2(D)$ where $\partial L/\partial W$ is the gradient and $s_m(D)$ is the minimum singular value of the extended feature matrix D (defined in Section 6.3.1). The minimum singular value is lower bounded by two terms, where the first term is related to the spectrum prop-

erty of the kernel function associate with the activation $\sigma(\cdot)$, and the second term quantifies the diversity of the units, measured by the classical notion of geometric discrepancy of a set of vectors. Essentially, the slower the decay of the spectrum, the better the optimization landscape; the more diverse the unit weights, the more likely stationary points will result in small training loss and generalization error.

We bypass the hurdle of non-convexity by directly analyzing the first order optimality condition of the learning problem, which implies that there are no spurious local minima if the minimum singular value of the extended feature matrix is large enough. Bounding the singular value is challenging because it entangles the nonlinear activation function, the weights and data in a complicated way. Unlike most previous attempts, we directly analyze the effect of nonlinearity without assuming independence of the activation patterns from actual data; in fact, the dependence of the patterns on the data and the unit weights underlies the key connection to activation kernel spectrum and the diversity of the units.

Our results also suggest a novel regularization scheme to promote unit diversity for potentially better generalization. In [111], it is shown that diversity of the neurons leads to smaller network size and better performance.

Whenever possible, we corroborate our theoretical analysis with numerical simulations. These numerical results include computing and verifying the relationship between the discrepancy of a learned neural network and the minimum singular value. Additionally, we measure the effects on the discrepancy with and without regularization. In all these examples, the experiments match with the theory nicely and they accord with the practice of using gradient descent to learn neural networks.

6.2 Related work

Kernel methods have many commonalities with one-hidden-layer neural networks. The random feature perspective [55, 74] views kernels as linear combinations of nonlinear basis functions, similar to neural networks. The difference between the two is that the weights are

random in kernels while in neural networks they are learned. Using learned weights leads to considerable smaller models as shown in [109]. However it is a non-convex problem and it is difficult to find the global optima. *e.g.*, one-hidden-layer networks are NP-complete to learn in the worst case [112]. We will make novel use of techniques from kernel methods to analyze learning in neural networks.

The empirical success of training neural networks with simple algorithms such as gradient descent has motivated researchers to explain their surprising effectiveness. In [113], the authors analyze the loss surface of a special random neural network through spin-glass theory and show that for many large-size networks, there is a band of exponentially many local optima, whose loss is small and close to that of a global optimum. The analyzed polynomial network is different from the actual neural network being used which typically contains ReLU nowadays. Moreover, the analysis does not lead to a generalization guarantee for the learned neural network.

A similar work shows that all local optima are also global optima in linear neural networks [114]. However their analysis for nonlinear neural networks hinges on independence of the activation patterns from the actual data, which is unrealistic. Some other works try to argue that gradient descent is not trapped in saddle points [115, 116], as was suggested to be the major obstacle in optimization [117]. There is also a seminal work using tensor method to avoid the non-convex optimization problem in neural network [118]. However, the resulting algorithm is very different from typically used algorithms where only gradient information of the empirical loss L is used.

[119] is the closest to our work, which shows that zero gradient implies zero loss for all weights except an exception set of measure zero. However, this is insufficient to guarantee low training loss since small gradient can still lead to large loss.

Furthermore, their analysis does not characterize the exception set and it is unclear a priori whether the set of local minima fall into the exception set.

Some recent works [111, 120, 121] also focused on promoting diversity in neural net-

work weights however their results are not concerned with guaranteeing global optima.

6.3 Problem setting and preliminaries

We will focus on a special class of data distributions where the input $x \in \mathbb{R}^d$ is drawn uniformly from the unit sphere,[1] and assume that $|y| \leq Y$. We consider the following hypothesis class:

$$\mathcal{F} = \left\{ \sum_{k=1}^{n} v_k \sigma(w_k^\top x) : v_k \in \{\pm 1\}, \sum_{k=1}^{n} \|w_k\| \leq C_W \right\} \tag{6.3}$$

where $\sigma(\cdot) = \max\{0, \cdot\}$ is the rectified linear unit (ReLU) activation function, $\{w_k\}$ and $\{v_k\}$ are the unit weights and combination coefficients respectively, n is the number of units, and C_W is some constant. We restrict $v_k \in \{-1, 1\}$ due to the positive homogeneity of ReLU,

$$f(x) = \sum_{k=1}^{n} v_k \sigma(w_k^\top x) = \sum_{k=1}^{n} \frac{v_k}{|v_k|} \sigma(|v_k| w_k^\top x).$$

That is, the magnitude of v_k can always be scaled into the corresponding w_k.

For convenience, let

$$W := (w_1^\top, w_2^\top, \ldots, w_k^\top)^\top \tag{6.4}$$

be the column concatenation of the unit parameters; also let W denote the set $\{w_i : 1 \leq i \leq k\}$. Let

$$\mathcal{F}_W = \left\{ W : \sum_{k} \|w_k\| \leq C_W \right\} \tag{6.5}$$

[1] The restriction of input to the unit sphere is not too stringent since many inputs are already normalized. Furthermore, it is possible to relax the uniform distribution to sub-gaussian rotationally invariant distributions, but we use the current assumption for simplicity.

denote the feasible set of W's. A function $f \in \mathcal{F}$ will depend on v and W, and it can be written as $f(x; v, W)$. But when clear from the context, it is shorten as $f(x)$.

Given a set of *i.i.d.* training examples $\{x_l, y_l\}_{l=1}^m \subseteq \mathbb{R}^d \times \mathbb{R}$, we want to find a function $f \in \mathcal{F}$ which minimizes the following least-squares loss function [2]

$$L(f) = \frac{1}{2m} \sum_{l=1}^{m} \left(y_l - f(x_l)\right)^2 .$$

Typically, gradient descent over $L(f)$ is used to learn all the parameters in f, and a solution with small gradient is returned at the end.[3] However, adjusting the bases $\{w_k\}$ leads to a non-convex optimization problem, and there is no theoretical guarantee that gradient descent can find global optima.

Our primary goal is to identify conditions under which there are no spurious local minima. We need to identify a set $\mathcal{G}_W \subseteq \mathcal{F}_W$ such that when gradient descent outputs a solution $W \in \mathcal{G}_W$ with the gradient norm $\|\partial L/\partial W\|$ smaller than ϵ, then the training and test errors can be bounded by $O(\epsilon^2)$. Ideally, $\mathcal{G}_W$ should have clear characterization that can be easily verified, and should contain most W in the parameter space (especially those solutions obtained in practice).

On notation, we will use c, c' or C, C' to denote constants and its value may change from line to line.

6.3.1 First order condition

In this section, we will rewrite the set of first order conditions for minimizing the empirical loss L. This rewriting motivates the direction of our later analysis. More specifically, the

[2] Our approach can also be applied to some other loss functions. For example, in logistic regression for $y \in \{\pm 1\}$, when the loss is $\frac{-1}{m} \sum_{\ell=1}^{m} \log\left[\text{sigmoid}(y_\ell f(x_\ell))\right]$, we can bound the error $\mathbb{E}\left[\text{sigmoid}(yf(x)) - 1\right]^2$.

[3] Note that even though ReLU is not differentiable, we can still use its sub-gradient by defining $\sigma'(u) = \mathbb{I}\left[u > 0\right]$.

gradient of the empirical loss w.r.t. w_k is

$$\frac{\partial L}{\partial w_k} = \frac{1}{m}\sum_{l=1}^{m}\left(f(x_l) - y_l\right) v_k \sigma'(w_k^\top x_l) x_l, \tag{6.6}$$

for all $k = 1, \ldots, n$. We will express this collection of gradient equations using matrix notation. Define the "extended feature matrix" as

$$D = \begin{pmatrix} v_1\sigma'(w_1^\top x_1)x_1 & \cdots & v_1\sigma'(w_1^\top x_m)x_m \\ \cdots & \cdots & \cdots \\ v_k\sigma'(w_k^\top x_1)x_1 & \cdots & v_k\sigma'(w_k^\top x_m)x_m \\ \cdots & \cdots & \cdots \\ v_n\sigma'(w_n^\top x_1)x_1 & \cdots & v_n\sigma'(w_n^\top x_m)x_m \end{pmatrix},$$

and the residual as

$$r = \frac{1}{m}\left(f(x_1) - y_1, \cdots, f(x_m) - y_m\right)^\top.$$

Then we have

$$\frac{\partial L}{\partial W} := \left(\frac{\partial L}{\partial w_1}^\top, \ldots, \frac{\partial L}{\partial w_n}^\top\right)^\top = D\, r. \tag{6.7}$$

A stationary point has zero gradient, so if $D \in \mathbb{R}^{dn\times m}$ has full column rank, then immediately $r = 0$, *i.e.*, it is actually a global optimal. Since $nd > m$ is necessary for D to have full column rank, we assume this throughout this chapter.

However, in practice, we will not have the gradient being exactly zero because, *e.g.*, we stop the algorithm in finite steps or because we use stochastic gradient descent (SGD). In other words, typically we only have $\|\partial L/\partial W\| \le \epsilon$, and D being full rank is insufficient since small gradient can still lead to large loss. More specifically, let $s_m(D)$ be the minimum singular value of D, we have

$$\|r\| \leq \frac{1}{s_m(D)} \left\| \frac{\partial L}{\partial W} \right\|. \tag{6.8}$$

We can see that $s_m(D)$ needs to be large enough for the residual to be small. Thus it is important to identify conditions to lower bound $s_m(D)$ away from zero, which will be the focus of this chapter.

6.3.2 Spectrum decay of activation kernel

We will later show that $s_m(D)$ is related to the decay rate of the kernel spectrum associated with the activation function. More specifically, for an activation function $\sigma(w^\top x)$, we can define the following kernel function

$$g(x, y) = \mathbb{E}_w[\sigma'(w^\top x)\sigma'(w^\top y)\langle x, y\rangle] \tag{6.9}$$

where $\mathbb{E}_w$ is over w uniformly distributed on a sphere.

In particular, for ReLU, the kernel has a closed form

$$g(x, y) = \left(\frac{1}{2} - \frac{\arccos\langle x, y\rangle}{2\pi}\right)\langle x, y\rangle . \tag{6.10}$$

In fact, it is a dot-product kernel and its spectrum can be obtained through spherical harmonic decomposition:

$$g(x, y) = \sum_{u=1}^{\infty} \gamma_u \phi_u(x)\phi_u(y), \tag{6.11}$$

where the eigenvalues are ordered $\gamma_1 \geq \cdots \geq \gamma_m \geq \cdots \geq 0$ and the bases $\phi_u(x)$ are spherical harmonics. The m-th eigenvalue γ_m will be related to $s_m(D)$.

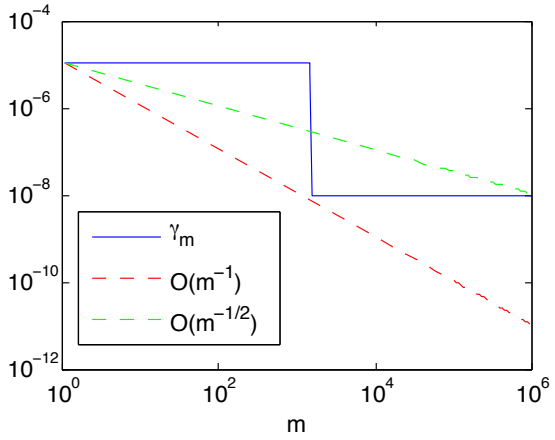


Figure 6.1: The spectrum decay of the kernel associated with ReLU. We set $d = 1500$. It is decays slower than $O(1/m)$ for a large range of m.

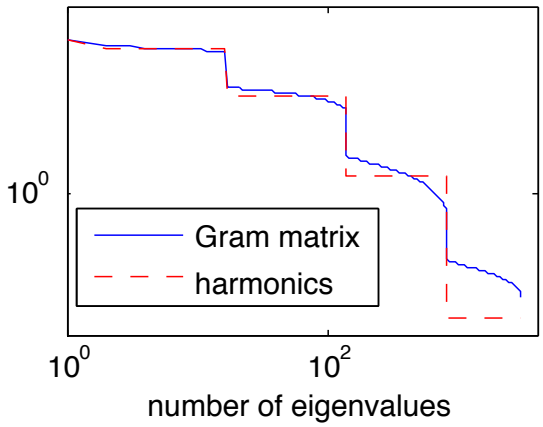


Figure 6.2: The spectrum of a Gram matrix concentrates around the spherical harmonic spectrum of the kernel.

For each spherical harmonic of order t, there are $N(d,t) = \frac{2t+d-2}{t}\begin{pmatrix} t+d-3 \\ t-1 \end{pmatrix}$ basis functions sharing the same eigenvalue. Therefore, the spectrum has a step like shape where each step is of length $N(d,t)$. Especially, for high dimensional input x, the number of such basis functions with large eigenvalues can be very large. Figure 6.1 illustrates the spectrum of the kernel for $d = 1500$, and it is about $\Omega(m^{-1})$ for a large range of m. For more details about the decomposition, please refer to Appendix C.1.

Such step like shape also appears in the Gram matrix associated with the kernel. Figure 6.2 compares the spectra of the kernel of $d = 15$ and the corresponding Gram matrix with $m = 3000$. We can see the spectrum of the Gram matrix closely resembles that of the kernel. Such concentration phenomenon underlies the reason why the spectrum of $D^\top D$ is closely related to the corresponding kernel.

6.3.3 Weight discrepancy

Another key factor in the analysis is the diversity of the unit weights, measured by the classic notion of geometric discrepancy [122]. Given a set of n points $W = \{w_k\}_{k=1}^n$ on

the unit sphere $\mathbb{S}^{d-1}$, the discrepancy of W w.r.t. a measurable set $S \subseteq \mathbb{S}^{d-1}$ is defined as

$$\text{dsp}(W, S) = \frac{1}{n} |W \cap S| - A(S), \tag{6.12}$$

where $A(S)$ is the normalized area of S (i.e., the area of the whole sphere $A(\mathbb{S}^{d-1}) = 1$). $\text{dsp}(W, S)$ quantifies the difference between the empirical measure of S induced by W and the measure of S induced by a uniform distribution.

By defining a collection $\mathcal{S}$ of such sets, we can summarize the difference in the empirical measure induced by W versus the uniform distribution over the sphere. More specifically, we will focus on the set of slices, each defined by a pair of inputs $x, y \in \mathbb{S}^{d-1}$, *i.e.*,

$$\begin{aligned} \mathcal{S} &= \left\{ S_{xy} : x, y \in \mathbb{S}^{d-1} \right\}, \text{ where} \\ S_{xy} &= \left\{ w \in \mathbb{S}^{d-1} : w^\top x \geq 0, w^\top y \geq 0 \right\}. \end{aligned} \tag{6.13}$$

Essentially, each S_{xy} defines a slice-shaped area on the sphere which is carved out by the two half spaces $w^\top x \geq 0$ and $w^\top y \geq 0$.

Based on the collection $\mathcal{S}$, we can define two discrepancy measures relevant to ReLU units. L_∞ discrepancy of W w.r.t. $\mathcal{S}$ is defined as

$$L_\infty(W, \mathcal{S}) = \sup_{S \in \mathcal{S}} \left| \text{dsp}(W, S) \right|, \tag{6.14}$$

and the L_2 discrepancy as

$$L_2(W, \mathcal{S}) = \sqrt{\mathbb{E}_{x,y} \text{dsp}(W, S_{xy})^2} \tag{6.15}$$

where the expectation is taken over x, y uniformly on the sphere. We use $L_\infty(W)$ and $L_2(W)$ as their shorthands. Both discrepancies measure how diverse the points W are. The

more diverse the points, the smaller the discrepancy.

For our analysis, we slightly generalize the discrepancy for w_k's not on unit sphere, by setting

$$\text{dsp}(\{w_k\}_k, S) = \text{dsp}(\{w_k/\|w_k\|\}_k, S). \tag{6.16}$$

6.4 Main results

Our main result is a bound on the training loss and the generalization error, assuming sufficiently large n, d. To state the theorem, first recall that $\beta \in (0,1)$ is the decay exponent of the spectrum of the activation kernel in (6.10), that is, γ_m is the m-th eigenvalue of the kernel and satisfies $\gamma_m = \Omega(m^{-\beta})$. Also recall that $\mathcal{F}_W$ denote the set of feasible values of W.

Theorem 24 (Main, simplified) *Let $\delta, \delta' \in (0,1)$. If*

$$n = \tilde{\Omega}(m^\beta), \quad d = \tilde{\Omega}(m^\beta),$$

then there exists a set $\mathcal{G}_W \subseteq \mathcal{F}_W$ which takes up $1-\delta'$ fraction of measure of $\mathcal{F}_W$, such that with probability at least $1 - cm^{-\log m} - \delta$ the following holds. For any $W \in \mathcal{G}_W$ and any $v \in \{-1,1\}^n$, we have

$$\frac{1}{2m}\sum_{\ell=1}^{m}(f(x_\ell; v, W) - y_\ell)^2 \le c\left\|\frac{\partial L}{\partial W}\right\|^2,$$
$$\frac{1}{2}\mathbb{E}(f(x; v, W) - y)^2 \le c\left\|\frac{\partial L}{\partial W}\right\|^2 + c'(C_W + Y)^2\sqrt{\frac{1}{m}\log\frac{1}{\delta}}.$$

Here $\tilde{\Omega}$ hides logarithmic terms $\log m \log\frac{1}{\delta}\log\frac{1}{\delta'}$.

Remark 1. Intuitively, the theorem means that when n, d are sufficiently large, for most feasible weights W, the training loss is in the order of the square norm of the gradient, and

the generalization error has an additional term $\tilde{O}(1/\sqrt{m})$. In particular, when we obtain an solution $W \in \mathcal{G}_W$ with gradient $\|\partial L/\partial W\|^2 \leq \epsilon$, the training loss is bounded by $O(\epsilon)$, and the generalization error is $\tilde{O}(\epsilon + 1/\sqrt{m})$. So neural networks trained with sufficiently many data points to a solution with small gradient in $\mathcal{G}_W$ generalize well. This essentially means that although non-convex, the loss function over this set is well behaved, and thus learning is not difficult.

Note that an immediate corollary is that any critical point in $\mathcal{G}_W$ is global optimum.

Furthermore, a randomly sampled set of weights W are likely to fall into this set. This suggests a reason for the practical success of training with random initialization: after initialization, the parameters w.h.p. fall into the set, then would stay inside during training, and finally get to a point with small gradient, which by our analysis, has small error.

Remark 2. An important feature about our result is that the set $\mathcal{G}_W$ has a simple explicit form:

$$\mathcal{G}_W = \left\{ W \in \mathcal{F}_W : (L_2(W))^2 \leq c_g \left(\sqrt{\frac{\log d}{nd} \log \frac{1}{\delta'}} + \frac{1}{n} \log \frac{1}{\delta'} \right) \right\} \tag{6.17}$$

where $c_g > 0$ is a universal constant. Furthermore, $(L_2(W))^2$ has a simple closed form (See Theorem 30). Therefore, it is possible to directly check if a solution W is in $\mathcal{G}_W$, or design regularization that make sure W stays in the set $\mathcal{G}_W$.

Remark 3. The above theorem is a special case of the following more general result.

Theorem 25 (Main, general) *Let $\delta \in (0, 1)$. With probability $\geq 1 - m \exp(-m\gamma_m/8) - 2m^2 \exp\left(-4 \log^2 d\right) - \delta$, the following holds. For any $\xi, \eta > 0$ and any W with $L_2(W) = \tilde{O}\left(n^{-\xi} d^{-\eta}\right)$ such that*

$$m^\beta \leq \tilde{O}\left\{ d^{(1+\eta)/2} n^{\xi/2} m^{1/4}, d^{1/2} m^{1/2}, n^\xi d^{1/2+\eta} \right\}, \tag{6.18}$$

and any $v \in \{-1,1\}^n$*, we have*

$$\frac{1}{2m}\sum_{\ell=1}^{m}(f(x_\ell;v,W)-y_\ell)^2 \le \frac{cm^\beta}{n}\left\|\frac{\partial L}{\partial W}\right\|^2,$$
$$\frac{1}{2}\mathbb{E}(f(x;v,W)-y)^2 \le \frac{cm^\beta}{n}\left\|\frac{\partial L}{\partial W}\right\|^2 + c'(C_W+Y)^2\sqrt{\frac{1}{m}\log\frac{1}{\delta}}.$$

Here $\tilde{\Omega}$ hides logarithmic terms $\log m \log\frac{1}{\delta}\log\frac{1}{\delta'}$.

The theorem shows that when the weights are diverse (*i.e.*, with good discrepancy $L_2(W)$ so that ξ and η are sufficiently large), the training loss is proportional to m^β/n times the squared norm of the gradient. This implies a local minimum leads to a global minimum and the neural network learns the target function. The generalization error has an additional term $\tilde{O}\left(1/\sqrt{m}\right)$.

To obtain Theorem 24 from this more general result, we first note that Lemma 31 proves most W falls into $\mathcal{G}_W$, so it is sufficient to choose n, d large enough to guarantee the condition (6.18). Setting $n = \tilde{O}(m^\beta)$ and $d = \tilde{O}(m^\beta)$ satisfies the condition with $\xi = \eta = 1/4$, and thus leads to Theorem 24. Clearly, there exist some other options. For example, $d = \tilde{\Omega}\left(m^{2\beta-1}\right)$ and $n = \tilde{\Omega}\left(m^{3-2\beta}\right)$, which matches the empirical observation that overspecified networks with large n are easier for the optimization.

6.5 Analysis roadmap

Our key technical result is a lower bound on the smallest singular value of D based on the spectrum of the activation kernel defined in (6.10) and the discrepancy of the weights defined in (6.15). Once the lower bound is obtained, we can use (6.8) to bound the training loss, and use Rademachar complexity to bound the generalization error.

Theorem 26 (Smallest singular value) *The following holds with probability* $\ge 1-\delta-m\exp\left(-m\gamma_m/8\right)-2m^2\exp\left(-4\log^2 d\right)$*. For any* $\xi,\eta > 0$*, any* W *with* $L_2(W) =$

$\tilde{O}\left(n^{-\xi}d^{-\eta}\right)$, *and any* $v \in \{-1,1\}^n$,

$$s_m(D)^2 \geq \Omega(nm^{1-\beta}) - \tilde{O}\left(\frac{n^{1-\xi/2}m^{3/4}}{d^{(1+\eta)/2}}\right) - \tilde{O}\left(\frac{nm^{1/2}}{d^{1/2}}\right) - \tilde{O}\left(\frac{n^{1-\xi}m}{d^{1/2+\eta}}\right).$$

Here the notation $\tilde{\Omega}$ hides logarithmic terms $\log d \log \frac{1}{\delta}$.

The theorem is stated in its general form. It bounds the smallest singular value in terms of the n, d, m and two parameters ξ, η quantifying how large $L_2(W)$ is. It is instructive to consider the special case when $n = \tilde{\Omega}(m^\beta)$, $d = \tilde{\Omega}(m^\beta)$, and $\xi = \eta = 1/4$, which corresponds to Theorem 24). In this case, with probability at least $1 - cm^{-\log m} - \delta$, $s_m(D)^2 \geq cm$ for some constant $c > 0$. It is clear that the singular value is large and bounded away from zero.

It is interesting to compare the theorem to the results in [119], which shows that D is full rank with probability one under small perturbations. However, full-rankness alone is not sufficient since its smallest singular value could be extremely small leading to possibly huge training loss. Instead, we directly bound the smallest singular value and relate it to the activation and the diversity of the weights.

Intuition. Here we describe the high level intuition for bounding the minimum singular value. It is necessarily connected to the activation function and the diversity of the weights. For example, if $\sigma'(t)$ is very small for all t, then the smallest singular value is expected to be very small. For the weights, if $d < m$ (the interesting case) and all w_k's are the same, then D cannot have rank m. If w_k's are very similar to each other, then one would expect the smallest singular value to be very small or even zero. Therefore, some notion of diversity of the weights is needed.

The analysis begins by considering the matrix $G_n = D^\top D/n$. It suffices to bound $\lambda_m(G_n)$, the m-th (and the smallest) eigenvalue of G_n. To do so, we introduce a matrix G whose entries $G(i,j) = \mathbb{E}_w[G_n(i,j)]$ where the expectation $\mathbb{E}_w$ is taken assuming w_k's are uniformly random on the unit sphere. The intuition is that when w is uniformly distributed,

$\sigma'(w^\top x)$ is most independent of the actual value of the x, and the matrix D should have the highest chance of having large smallest singular value. We will introduce G as an intermediate quantity and subsequently bound the spectral difference between G_n and G.

Roughly speaking, we break the proof into two steps

$$\lambda_m(G_n) \geq \underbrace{\lambda_m(G)}_{\text{I. ideal spectrum}} - \underbrace{\|G - G_n\|}_{\text{II. discrepancy}}$$

where $\|G - G_n\|$ is the spectral norm of the difference.

For the first term in the lower bound, we observe that G has a particular nice form: $G(i,j) = g(x_i, x_j)$, the kernel defined in (6.10). This allows us to apply the eigendecomposition of the kernel and positive definite matrix concentration inequality to bound $\lambda_m(G)$, which turns out to be around $m\gamma_m/2$.

For the second term, when w_k's are indeed from the uniform distribution over the sphere, this can be bounded by concentration bounds. It turns out that when w_k's are not too far away from that, it is still possible to do so. Therefore, we use the geometric discrepancy to measure the diversity of the weights, and show that when they are sufficiently diverse, $\|G - G_n\|$ is small. In particular, the entries in $G - G_n$ can be viewed as the kernel of some U-statistics, hence concentration bounds can be applied. The expected U-statistics turns out to be the $(L_2(W))^2$, which has a closed form and can be shown to be small.

Outline. Theorem 26 is proved in Section 6.6, $L_2(W)$ and $\mathcal{G}_W$ are characterized in Section 6.6.2, and the proof sketch of Theorem 24 and Theorem 25 is provided in Section 6.7. We describe the proof sketch for the lemmas and provide the remaining proofs in the appendix.

6.6 Bounding the smallest singular value

Theorem 26 can be obtained from the following technical lemma.

Lemma 27 *With probability* $\geq 1 - m\exp\left(-m\gamma_m/8\right) - 2m^2\exp\left(-4\log^2 d\right) - \delta$*, we have*

$$s_m(D)^2 \geq nm\gamma_m/2 - cn\rho(W), \tag{6.19}$$

where

$$\begin{aligned}\rho(W) =& \frac{\log d}{\sqrt{d}}\sqrt{L_\infty(W)L_2(W)}m\left(\frac{4}{m}\log\frac{1}{\delta}\right)^{1/4} \\ &+ \frac{\log d}{\sqrt{d}}mL_\infty(W)\sqrt{\frac{4}{3m}\log\frac{1}{\delta}} + \frac{\log d}{\sqrt{d}}mL_2(W) + L_\infty(W).\end{aligned} \tag{6.20}$$

Proof [Proof of Theorem 26] First, $|\text{dsp}(W,S)| \leq 2$ for any set W and slice S, so by definition $|L_\infty(W)| \leq 2$. Next, By the assumption in the theorem, $L_2(W) = \tilde{O}(n^{-\xi}d^{-\eta})$. Plugging these into Lemma 27 completes the proof. ■

Lemma 27 is meaningful only when $cn\rho(W)$ is small compared to $nm\gamma_m/2$. This requires $L_2(W)$ to be sufficiently small. In the following we will first provide the proof sketch of Lemma 27, and then bound that $L_2(W)$ in Section 6.6.2.

6.6.1 Proof of Lemma 27

To prove Lemma 27, it is sufficient to bound the smallest eigenvalue of $G_n = D^\top D/n$. Note that $v_k \in \{-1, 1\}$, so $v_k^2 = 1$, and thus the (i,j)-th entry of G_n is

$$G_n(i,j) = \frac{1}{n}\sum_{k=1}^{n}\sigma'(w_k^\top x_i)\sigma'(w_k^\top x_j)\left\langle x_i, x_j\right\rangle. \tag{6.21}$$

For ReLU, $\sigma'(w^\top x)$ does not depend on the norm of w so without loss of generality, we assume $\|w\| = 1$. Consider a related matrix G whose (i,j)-th entry is defined as

$$G(i,j) = \mathbb{E}_w\left[\sigma'(w^\top x_i)\sigma'(w^\top x_j)\left\langle x_i, x_j\right\rangle\right]. \tag{6.22}$$

where w is a random variable distributed uniformly over the unit sphere. Since $\sigma'(w^\top x) =$

$\mathbb{I}\left[w^\top x \geq 0\right]$, we have a closed form expression for $G(i,j)$:

$$\begin{aligned} G(i,j) &= \mathbb{E}_w\left[\mathbb{I}(w^\top x_i \geq 0)\mathbb{I}(w^\top x_j \geq 0)\right]\langle x_i, x_j\rangle \\ &= \left(\frac{1}{2} - \frac{\arccos\langle x_i, x_j\rangle}{2\pi}\right)\langle x_i, x_j\rangle . \end{aligned} \tag{6.23}$$

Note that $G(i,j) = g(x_i, x_j)$ where g is the kernel defined in (6.10). This allows us to reason about the eigenspectrum of G, denoted as $\lambda_1(G) \geq \ldots \geq \lambda_m(G)$.

Therefore, our strategy is to first bound $\lambda_m(G)$ in Lemma 28 and then bound $|\lambda_m(G) - \lambda_m(G_n)|$ in Lemma 29. Combining the two immediately leads to Lemma 27.

First, consider $\lambda_m(G)$. We consider a truncated version of spherical harmonic decomposition:

$$g^{[m]}(x_i, x_j) = \sum_{u=1}^{m} \gamma_u \phi_u(x_i)\phi_u(x_j)$$

and the corresponding matrix $G^{[m]}$. On one hand, it is clear that $\lambda_m(G) \geq \lambda_m(G^{[m]})$. On the other hand, $G^{[m]} = AA^\top$ where A is a random matrix whose rows are

$$A^i := [\sqrt{\gamma_1}\phi_1(x_i), \ldots, \sqrt{\gamma_m}\phi_m(x_i)].$$

Next, we bound $\lambda_m(G^{[m]})$ by matrix Chernoff bound [123], and it is better than previous work [124]. This leads to the following lemma.

Lemma 28 *With probability at least* $1 - m\exp\left(-m\gamma_m/8\right)$,

$$\lambda_m(G) \geq m\gamma_m/2.$$

Next, bound $|\lambda_m(G) - \lambda_m(G_n)|$. By Weyl's theorem, this is bounded by $\|G - G_n\|$. To simplify the notation, denote

$$E_{i,j} = \mathbb{E}_w[\sigma'(w^\top x_i)\sigma'(w^\top x_j)] \; - \frac{1}{n}\sum_{k=1}^{n} \sigma'(w_k^\top x_i)\sigma'(w_k^\top x_j).$$

Then $G(i,j) - G_n(i,j) = \langle x_i, x_j \rangle E_{ij}$, and thus

$$\begin{aligned}
&\|G - G_n\| \\
&= \sup_{\|\alpha\|=1} \sum_{i,j} \alpha_i \alpha_j \langle x_i, x_j \rangle E_{ij} \\
&\leq \sup_{\|\alpha\|=1} \sqrt{\sum_{i \neq j} \alpha_i^2 \alpha_j^2} \sqrt{\sum_{i \neq j} \langle x_i, x_j \rangle^2 E_{ij}^2} + \max_i |E_{ii}| \\
&\leq \frac{c \log d}{\sqrt{d}} \sqrt{\sum_{i \neq j} E_{ij}^2} + \max_i |E_{ii}| \qquad (6.24)
\end{aligned}$$

where the last inequality holds with high probability since x_i's are uniform over the unit sphere and thus we can apply sub-gaussian concentration bounds.

Note that $\sum_{i \neq j} E_{ij}^2/(m(m-1))$ is a U-statistics where the summands are dependent and typical concentration inequality for *i.i.d.* entries does not apply. Instead we use a Bernstein inequality for U-statistics [125] to show that with probability at least $1 - \delta$, it is bounded by

$$\mathbb{E}_{\{x_1,x_2\}} E_{12}^2 + \sqrt{\frac{4\Sigma^2}{m} \log \frac{1}{\delta}} + \frac{4B^2}{3m} \log \frac{1}{\delta} \qquad (6.25)$$

where $B = \max_i |E_{ii}|$ and $\Sigma^2 = \mathbb{E}\left[E_{12}^4\right] - \left(\mathbb{E}\left[E_{12}^2\right]\right)^2$.

The key observation is that the quantities in the above lemma are related to discrepancy:

$$\max_{i,j} E_{ij} \leq L_\infty(W), \qquad (6.26)$$

$$\mathbb{E}_{x_1,x_2}\left[E_{12}^2\right] = (L_2(W))^2, \qquad (6.27)$$

$$\Sigma^2 \leq (L_2(W) L_\infty(W))^2. \qquad (6.28)$$

This is because $\sigma'(w^\top x_i)\sigma'(w^\top x_j) = \mathbb{I}[w \in S_{x_i x_j}]$ and thus

$$\begin{aligned} E_{i,j} &= \mathbb{E}_w \mathbb{I}[w \in S_{x_i x_j}] - \frac{1}{n}\sum_{k=1}^{n} \mathbb{I}[w_k \in S_{x_i x_j}] \\ &= A(S_{x_i x_j}) - \frac{1}{n}\left|W \cap S_{x_i x_j}\right| \\ &= -\text{dsp}(W, S_{x_i x_j}). \end{aligned}$$

Plugging (6.26)-(6.28) into (6.25) and (6.24), we have

Lemma 29 *The following inequality holds with probability at least* $1-2m^2\exp\left(-\log^2 d\right)-\delta$*,*

$$\|G_n - G\| \leq c\rho(W) \tag{6.29}$$

where $\rho(W)$ *is as defined in Lemma 27.*

6.6.2 Characterizing the discrepancy

In this subsection, we present a bound for $L_2(W)$ and show that the $\mathcal{G}_W$ defined in the following covers most W's. Recall that

$$\mathcal{G}_W = \left\{ W : (L_2(W))^2 \leq c_g \left(\sqrt{\frac{\log d}{nd}\log\frac{1}{\delta'}} + \frac{1}{n}\log\frac{1}{\delta'} \right) \right\}$$

for $0 < \delta' < 1$ and a proper constant $c_g > 0$. The constant c_g is the constant in Lemma 31. δ' will be clear from the context where $\mathcal{G}_W$ is used.

First we provide a closed form for L_2 discrepancy of slices defined in (6.13). The proof is provided in the appendix.

Theorem 30 *Suppose* $W = \{w_i\}_{i=1}^n \subseteq \mathbb{S}^{d-1}$.

$$(L_2(W))^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} k(w_i, w_j)^2 - \mathbb{E}_{u,v}\left[k(u,v)^2\right]$$

where $\mathbb{E}_{u,v}$ is over u and v uniformly distributed on $\mathbb{S}^{d-1}$ and the kernel $k(\cdot,\cdot)$ is

$$k(u,v) = \frac{\pi - \arccos\langle u, v\rangle}{2\pi}.$$

The closed form is simple and intuitive. The kernel $k(w_i, w_j)$ measures how similar two units are. The discrepancy is the difference between the average pairwise similarity and the expected one over uniform distribution.

Given the theorem, we now show that $(L_2(W))^2$ can be small. We use the probabilistic method, *i.e.*, show that if w_k's are sampled from $\mathbb{S}^{d-1}$ uniformly at random, then with high probability W falls into $\mathcal{G}_W$. The key observation is that with random W, $(L_2(W))^2$ is the difference between a U-statistics and its expectation, which can be bounded by concentration inequalities. Formally,

Lemma 31 *There exists a constant c_g, such that for any $0 < \delta' < 1$, with probability at least $1 - \delta'$ over $W = \{w_i\}_{i=1}^n$ that are sampled from the unit sphere uniformly at random,*

$$(L_2(W))^2 \le c_g \left(\sqrt{\frac{\log d}{nd} \log \frac{1}{\delta'}} + \frac{1}{n} \log \frac{1}{\delta'} \right).$$

Alternatively, the theorem means that $\mathcal{G}_W$ defined in (6.17) covers most W's. This is because $L_2(W)$ is independent of the length of w_k's, it is sufficient to show that $L_2(W)$ is small for $W \in \mathcal{G}_W \cap \mathbb{S}^{d-1}$.

6.7 Final bound on generalization error

Here we provide the proof sketch of Theorem 25 and Theorem 24. More details of the proof are in Appendix C.6.

First, we prove Theorem 25. Suppose a solution W satisfies the assumption and has small gradient $\|\partial L/\partial W\|$. Using (6.8), we have $\|r\| \le \|\partial L/\partial W\|/s_m(D)$. By Theorem 26 and the assumption in Theorem 25, with high probability $s_m^2(D) = \Omega(nm^{1-\beta})$.

This implies the training loss is

$$\frac{1}{m}\sum_{l}\left(f(x_l)-y_l\right)^2 = m\left\|r\right\|^2 \leq \frac{cm^{\beta}}{n}\left\|\frac{\partial L}{\partial W}\right\|^2.$$

The generalization error can then be derived using McDiamid's inequality and Rademacher complexity. First, we need an upper bound on the difference of the loss for two data points for the McDiamid's inequality. Since $\|x\|_2 \leq 1$ and $\sum_k \|w_k\|_2 \leq C_W$, we have $|f| \leq C_W$. Thus

$$\begin{aligned}&|\ell(y,f(x))-\ell(y',f(x'))|\\ \leq\ &\frac{1}{2}\max\left\{(y-f(x))^2,(y'-f(x'))^2\right\} \leq Y^2+C_W^2\end{aligned}$$

where in the last inequality we use the fact that the true function $|y| \leq Y$. Next, we use the composition rules to compute the Rademacher complexity. Since the complexity of linear functions $\left\{w^{\top}x : \|w\|_2 \leq b_W, \|x\|_2 \leq 1\right\}$ is $b_W/\sqrt{m}$ and $\sigma(\cdot)$ is 1-Lipschitz, and $\sum_k \|w_k\|_2 \leq C_W$, the complexity $\mathcal{R}_m(\mathcal{F}) \leq C_W/\sqrt{m}$. Composing it with the loss function, and applying the bound in [126], we get the final generalization bound.

To obtain Theorem 24 from the more general Theorem 25, we first note that Lemma 31 proves most W falls into $\mathcal{G}_W$, and setting $n = \tilde{O}(m^{\beta})$ and $d = \tilde{O}(m^{\beta})$ makes sure that any $W \in \mathcal{G}_W$ has $L_2(W) = \tilde{O}(n^{-1/4}d^{-1/4})$ satisfying the condition (6.18).

6.8 Discussions

In this section, we discuss and remark on further considerations and possible extensions of our current analysis.

6.8.1 Other loss functions

Currently, our analysis is tied to the least squares loss $\ell(y, f(x)) = \frac{1}{2}(y - f(x))^2$. It is fairly straightforward to generalize it to any strongly convex loss function, such as logistic loss. Note that the final objective function is *not* convex due to the non-convexity in neural networks, but most loss functions used in practice are strongly convex w.r.t. $f(x)$. Under the new setting, the residual is then

$$r = \frac{1}{m}\left(\ell'(y_1, f(x_1)), \cdots, \ell'(y_m, f(x_m))\right)^\top.$$

According to our analysis, the norm of the residual $\|r\|$ will be bounded. This in turn implies each individual $\ell'(y_l, f(x_l))$ will be small. Since the loss function $\ell(y, f(x))$ is strongly convex, the loss itself will be small.

6.8.2 Other activation functions

We can consider a family of activation functions of the form $\sigma(u) = \max\{u, 0\}^t$, *i.e.*, rectified polynomials [74, 127]. This requires two modifications to the analysis.

One is the corresponding kernel $k(x, y) = \mathbb{E}_w\left[\sigma'(w^\top x)\sigma'(w^\top y)\right]$ and $g(x, y) = k(x, y)\langle x, y\rangle$. When the input distribution is uniform, we can also compute the kernels in closed form as shown in [74]:

$$k_t(x, y) = \frac{J_{t-1}(\theta)}{2\pi} \tag{6.30}$$

where

$$J_t(\theta) = (-1)^t(\sin\theta)^{2t+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^t\left(\frac{\pi - \theta}{\sin\theta}\right), \tag{6.31}$$

and $\theta = \arccos\langle x, y\rangle$. Note that the subscript is $t - 1$ in (6.30) because we are computing on the derivative $\sigma'(u)$.

Examples for the first few t are listed as follows.

$$J_0(\theta) = \pi - \theta \tag{6.32}$$

$$J_1(\theta) = \sin\theta + (\pi - \theta)\cos\theta \tag{6.33}$$

$$J_2(\theta) = 3\sin\theta\cos\theta + (\pi - \theta)(1 + 2\cos^2\theta) \tag{6.34}$$

Larger t corresponds to more nonlinear activation functions and leads to slower decaying spectrum since there are more high frequency components.

We also need to change the definition of the discrepancy to accommodate the new kernels. Let

$$\begin{aligned}(L_2(W))^2 &= \mathbb{E}_{x_i,x_j}\left[\mathbb{E}_w[\sigma'(w^\top x_i)\sigma'(w^\top x_j)] \ - \frac{1}{n}\sum_{k=1}^{n}\sigma'(w_k^\top x_i)\sigma'(w_k^\top x_j)\right]^2 \\ &= \frac{1}{n^2}\sum_{i,j=1}^{n} k(w_i, w_j)^2 - \mathbb{E}_{u,v}\left[k(u,v)^2\right]. \end{aligned} \tag{6.35}$$

Therefore, the discrepancy is affected by how the kernels change due to change in activation functions.

The other modification is on the Rademacher complexity. Since the derivative $\sigma'(u) = t\max\{u, 0\}^{t-1}$, there is an additional factor of t in front of the complexity. That is, larger t leads to higher Rademacher complexity.

In summary, the best parameter t depends on the balance between the two conflicting effects. On one hand, larger t corresponds to slower decaying spectrum and makes the minimum singular value more likely to be larger. On the other hand, smaller t leads to better generalization since the Rademacher complexity is smaller.

6.8.3 (Sub)gradient of the activation function

Throughout this chapter, we have used one particular subgradient for the ReLU activation function: $\mathbb{I}[u > 0]$. At the point $u = 0$, there are many other valid subgradients as long

as its value is between 0 and 1. However, our analysis is not restricted to this particular subgradient. First of all, all the subgradients only differ at one point $u = 0$, which is of probability zero. Second, our analysis is probabilistic in nature. The first term in Lemma 27 is the expectation over W, which is insensitive to the probability zero event $u = 0$. The second term in Lemma 27 is related to $L_2(W)$, which is again expectation over all possible data, thus insensitive to the difference.

In summary, though for some $W \in \mathcal{G}_W$ the loss is not differentiable, one can define $\partial L/\partial W$ by using subgradients of ReLU σ as follows:

$$\sigma'(x) = \begin{cases} 0, & x < 0 \\ c, & x = 0 \\ 1, & x > 0 \end{cases} \tag{6.36}$$

for any $c \in [0, 1]$. Then under the conditions in our theorems, with high probability, for any $W \in \mathcal{G}_W$ and any definition of σ' in (6.36), the guarantees hold.

Other activation functions such as rectified polynomials are differentiable and thus they do not have such issue.

6.8.4 Other input distribution

When the input distribution is not uniform, the spectrum of the kernel function defined in (6.9) will be different because the spherical harmonic bases are defined with respect to the input distribution. To ensure the spectrum decays slowly, we need to find a corresponding distribution of W that "matches" the input distribution.

We provide some intuitions in finding such "matching" distribution. Suppose the input distribution is uniform on the set $E \in \mathbb{S}^{d-1}$, if a hyperplane whose normal is w does not "cut through" the set, then for all data points, they have the same sign $\mathbb{I}[w^\top x > 0]$. This will likely lead to rank deficiency in the extended feature matrix.

Table 6.1: Comparison of minimum eigenvalues with uniform and "matching" distributions. Note that the "matching" distribution corresponds to larger minimum eigenvalue for different dimensions. However, the difference becomes negligible when the dimension increases.

d	4	5	6	7
uniform	3.96×10^{-4}	0.0015	0.0032	0.0072
matching	$\mathbf{5.43 \times 10^{-4}}$	**0.0017**	0.0032	0.0072

Therefore, we prefer W to split the data points as much as possible. One such distribution of W is uniform on the set $F_E = \left\{w \in \mathbb{S}^{d-1} : \text{there exists } \ u \in E,\ \langle u, w\rangle = 0\right\}$. For example, if E is the intersection of the positive orthant and the unit sphere, $E = \left\{u \in \mathbb{S}^{d-1} : u_i \geq 0, \text{ for all } \ i \in [d]\right\}$, then the corresponding set F_E is the whole sphere excluding E and $-E$.

We have verified the phenomenon empirically. We have generated 3000 input data points uniform on the positive orthant E. We then compute the 3000×3000 Gram matrix, where the (i, j)-th entry is $\mathbb{E}_w \left[\sigma'(w^\top x_i)\sigma'(w^\top x_i) \langle x_i, x_j\rangle\right]$. The expectation is approximated by sampling 100,000 independent w's and then averaging. We compare two distributions of W: 1) uniform on the whole unit sphere; 2) uniform on F_E.

In Table 6.1, we compare the minimum eigenvalues with the two distributions. The uniform distribution on F_E always leads to larger or the same minimum eigenvalues. However, as dimension increases, the difference becomes negligible. Note that the difference between the uniform distribution on the whole sphere and uniform on F_E becomes exponentially small when the dimension d increases, because the proportion of E and $-E$ shrinks exponentially. This suggests that in high dimensions, uniform on the whole unit sphere is a reasonable distribution for W.

For a general input distribution $P(x)$, we can decompose it into small sets dx and on every set, the distribution is uniform with measure $P(x)dx$. Then every small sets corresponds to a distribution of W. The final distribution of W is the superposition of all such distributions, weighted by $P(x)dx$.

6.9 Numerical evaluation

In this section, we further investigate numerically the effects of gradient descent on the discrepancy and the effects of regularizing the weights using discrepancy measure.

6.9.1 Discrepancy and gradient descent

One limitation of the analysis is that we have not analyzed how to obtain a solution $W \in \mathcal{G}_W$ with small gradient. The theoretical analysis of gradient descent is left for future work. Meanwhile we provide some numerical results supporting our claims.

Although the set $\mathcal{G}_W$ contains most W's, it is still unclear whether the solutions given by gradient descent lie in the set. We design experiments to investigate this issue. The ground truth input data are of dimension $d = 50$ and true function consists of $n = 50$ units. We use networks of different n to learn the true function and perform SGD with batch size 100 and learning rate 0.1 for 5000 iterations. Figure 6.3 shows how $(L_2(W))^2$ changes as a function of n. It is slightly worse than $O(n^{-1})$ but scales better than $O(n^{-1/2})$, suggesting (stochastic) gradient descent outputs solutions with reasonable discrepancy.

6.9.2 Regularization

To reinforce solutions with small discrepancy, we propose a novel regularization term to minimize L_2 discrepancy:

$$R(W) = \frac{1}{n(n-1)} \sum_{i \neq j}^{n} k(w_i, w_j)^2. \tag{6.37}$$

It is essentially L_2 discrepancy without the constants.

To verify the effectiveness of the regularization term, we explore the relationship between the regularization and the minimum singular value. We first generate 20 random W's, all with $n = 100$ and $d = 100$, and compute their discrepancy and singular values using $m = 3000$. Then we optimize $R(W)$ and compare the quantities after optimization.

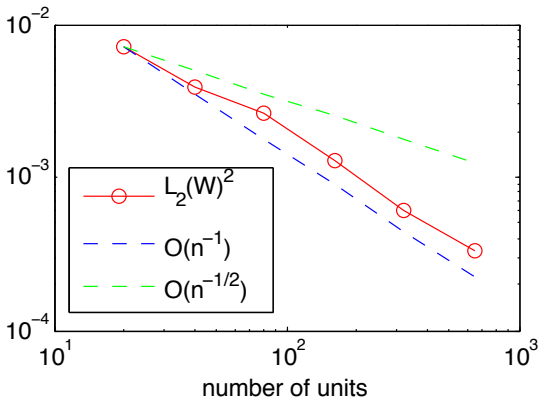


Figure 6.3: Discrepancy of W obtained after gradient descent. We perform gradient descent and compute discrepancy for the returned solution. The red curve corresponds to such solutions with different n. It scales similarly to the bound for uniform W as in Lemma 31.

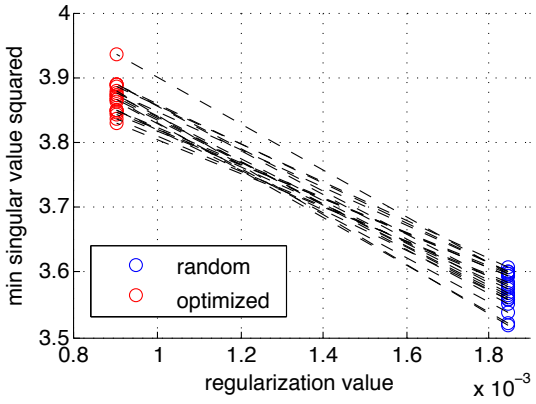


Figure 6.4: Effect of regularization. The blue dots represent random weights and the red dots linked with dashed black lines represent weights optimized by minimizing $R(w)$. Smaller regularization values correspond to larger minimal singular values.

The result is presented in Figure 6.4. We can see smaller regularization value corresponds to larger singular value.

We also conducts experiments to compare training and test errors with and without regularization. The ground truth data are of $d = 100$ and $n = 100$. We learn the true function by SGD with learning rate 0.1, momentum 0.9 and a total of 300,000 iterations. The regularization coefficients are chosen from $\{1, 0.1, 0.01, 0.001\}$ and the best results are reported. We use neural networks of size $n \in \{100, 150, 200, 300\}$ and for each n we repeat five times with different random seeds. The result is summarized in Table 6.2. Regularization leads to lower training and test errors for most settings. Even in the case where the un-regularized one performs better, the errors are all small enough (within the same range as standard deviation), suggesting the noise begins to dominate.

We also compare the regularization effects on the MNIST dataset. The dataset contains 60,000 training and 10,000 test handwritten digits. To demonstrate the regularization effect, we train one hidden layer fully connected neural networks with $k = 200, 400, 600, 800$ units. The results are summarized in Table 6.3. Note that state-of-the-arts performance on

Table 6.2: Comparison of performance with/without regularization (all numbers are of unit 10^{-5}). The true function is generated with $d = 100$ and $n = 100$. To learn the function, we use networks with different n.

	$n = 100$		$n = 150$	
	train	test	train	test
no-reg	15.42(5.86)	14.80(5.36)	1.79(0.45)	1.86(0.50)
reg	**11.32(1.77)**	**10.63(1.58)**	**1.07(0.84)**	**1.13(0.99)**
	$n = 200$		$n = 300$	
	train	test	train	test
no-reg	**0.38(0.27)**	**0.44(0.35)**	0.39(0.39)	0.44(0.40)
reg	0.50(0.51)	0.58(0.59)	**0.10(0.05)**	**0.12(0.07)**

Table 6.3: Performance comparison with/without regularization on MNIST dataset. Errors are all in $\%$.

	$n = 200$		$n = 400$	
	train	test	train	test
no-reg	0.94	3.39	**0.32**	3.08
reg	**0.56**	**3.22**	0.33	**2.90**
	$n = 600$		$n = 800$	
	train	test	train	test
no-reg	0.00065	2.67	0.11	2.90
reg	**0.00057**	**2.62**	**0.0003**	**2.45**

MNIST are mostly obtained by convolutional neural networks. This experiment is not intended to achieve the state-of-the-arts but it tries to showcase the advantage of regularization on a real-world dataset.

From Table 6.3, we see regularization consistently leads to slightly better test error for all cases.

6.10 Summary

We have analyzed one-hidden-layer neural networks and identified novel conditions when local optima become global optima despite the non-convexity of the loss function. The key factors are the spectrum of the kernel associated with the activation function and the diversity of the units measured by discrepancy.

CHAPTER 7

DEEP SEMI-RANDOM FEATURES

Inspired by the insight in the previous chapter, we propose semi-random features. Semi-random features are defined as the product of a random nonlinear switching unit and a linear adjustable unit. The flexibility of semi-random feature lies between the fully adjustable units in deep learning and the random features used in kernel methods. We show that semi-random features possess a collection of nice theoretical properties despite the non-convex nature of its learning problem. In experiments, we show that semi-random features can match the performance of neural networks by using slightly more units, and it outperforms random features by using significantly fewer units. Semi-random features provide an interesting data point in between kernel methods and neural networks to advance our understanding of the challenge of nonlinear function approximation, and it opens up new avenues to tackle the challenge further.

7.1 Introduction

There are two dominating paradigms for nonlinear modeling in machine learning: kernel methods and neural networks:

- Kernel methods employ *pre-defined* basis functions, $k(x, x')$, called kernels to represent nonlinear functions [75, 128]. Learning algorithms that use kernel methods often come with nice theoretical properties–globally optimal parameters can be found via convex optimization, and statistical guarantees can be provided rigorously. However, kernel methods typically work with matrices that are quadratic in the number of samples, leading to unfavorable computation and storage complexities. A popular approach to tackle such issues is to approximate kernel functions using random

features [61, 74, 129, 130, 131]. One drawback of random features is that its approximation powers suffer from the curse of dimensionality [109] because its bases are not adaptive to the data.

- Neural networks use adjustable basis functions and learn their parameters to approximate the target nonlinear function [132]. Such adaptive nature allows neural networks to be compact yet expressive. As a result, they can be efficiently trained on some of the largest datasets today. By incorporating domain specific network architectures, neural networks have also achieved state-of-the-art results in many applications. However, learning the basis functions involves difficult non-convex optimization. Few theoretical insights are available in the literature and more research is needed to understand the working mechanisms and theoretical guarantees for neural networks [133, 134, 135].

Can we have the best of both worlds? Can we develop a framework for big nonlinear problems which has the ability to adapt basis functions, has low computational and storage complexity, while at the same time retaining some of the theoretical properties of random features? Towards this goal, we propose semi-random features to explore the space of trade-off between flexibility, provability and efficiency in nonlinear function approximation. We show that semi-random features have a set of nice theoretical properties, like random features, while possessing a (deep) representation learning ability, like deep learning. More specifically:

- Despite the nonconvex learning problem, semi-random feature model with one hidden layer has no bad local minimum;
- Semi-random features can be composed into multi-layer architectures, and going deep in the architecture leads to more expressive model than going wide;
- Semi-random features also lead to statistical stable function classes, where generalization bounds can be readily provided.

Through experiments, using both large UCI datasets and image classification benchmarks such as MNIST, CIFAR10 and SVHN, we demonstrate the agreement between experiments and theoretical predictions.

An important contribution of our current study is the discovery of many interesting new insights to the problem, such as the structure of the optimization, the benefit of depth, and the tensorial inner product view, which advances our understanding of the problem and opens up new avenues to tackle the challenge further.

7.2 Background

We briefly review different ways of representing nonlinear functions in this section.

Hand-designed basis. In a classical machine learning approach for nonlinear function approximation, users or domain experts typically handcraft a set of features $\phi_{\text{expert}} : \mathcal{X} \to \mathcal{H}$, a map from an input data space $\mathcal{X}$ to a (complete) inner product space $\mathcal{H}$. Many empirical risk minimization algorithms then require us to compute the inner product of the features as $\langle \phi_{\text{expert}}(x), \phi_{\text{expert}}(x') \rangle_{\mathcal{H}}$ for each pair $(x, x') \in \mathcal{X} \times \mathcal{X}$. Computing this inner product can be expensive when the dimensionality of $\mathcal{H}$ is large, and indeed it can be infinite. For example, if $\mathcal{H}$ is the space of square integrable functions, we need to evaluate the integral as $\langle \phi_{\text{expert}}(x), \phi_{\text{expert}}(x') \rangle_{\mathcal{H}} = \int_{\omega} \phi_{\text{expert}}(x; \omega) \overline{\phi_{\text{expert}}(x'; \omega)}$.

Kernel methods. When our algorithms solely depend on the inner product, the kernel trick avoids this computational burden by introducing an easily computable kernel function as

$$k_{\text{expert}}(x', x) = \langle \phi_{\text{expert}}(x), \phi_{\text{expert}}(x') \rangle_{\mathcal{H}} ,$$

resulting in an implicit definition of the features ϕ_{expert} [75, 128]. However, the kernel approach typically scales poorly on large datasets. Given a training set of m input points $\{x_i\}_{i=1}^m$, evaluating a learned function at a new point x requires computing $\hat{f}(x) = \sum_{i=1}^m \alpha_i k_{\text{expert}}(x_i, x)$, the cost of which increases linearly with m. Moreover, it

usually requires computing (or approximating) inverses of matrices of size $m \times m$.

Random features. In order to scale to large datasets, one can approximate the kernel by a set of random basis functions sampled according to some distributions. That is,

$$k_{\text{expert}}(x', x) \approx \frac{1}{C} \sum_{j=1}^{C} \phi_{\text{random}}(x; \mathbf{r}_j) \phi_{\text{random}}(x'; \mathbf{r}_j),$$

where both the type of basis functions ϕ_{random}, and the sampling distribution for the random parameter $\mathbf{r}_j$ are determined by the kernel function. Due to its computational advantage and theoretical foundation, the random feature approach has many applications and is an active research topic [61, 74, 129, 130, 131].

Neural networks. Neural networks approximate functions using weighted combination of *adaptable* basis functions $f(x) = \sum_{k=1}^{n} w_k^{(2)} \phi(x; \mathbf{w}_k^{(1)})$, where both the combination weights $w_k^{(2)}$ and the parameters $\mathbf{w}_k^{(1)}$ in each basis function ϕ are learned from data. Neural networks can be composed into multilayers to express highly flexible nonlinear functions.

7.3 Semi-Random Features

When comparing different nonlinear representaitons, we can see that random features are designed to approximate a known kernel, but not for learning these features from the given dataset (i.e., it is not a *representation learning*). As a result, when compared to neural network, it utilizes less amount of information encoded in the dataset, which could be disadvantageous. Neural networks, on the other hand, pose a difficulty for theoretical developments due to non-convexity in optimization. However, a recent work of [114] showed that we can establish optimization theory despite its non-convexity, if we ignore or randomize the activation part of neural networks. In addition, [136] recently proved that the diverse (or nearly random) hidden weights leads to a good optimization (and generalization) bound for neural networks.

These insights suggest a hybrid approach of random feature and neural network, called

semi-random feature (or semi-random unit), to learn representation (or feature) from datasets. The goal is to obtain a new type of basis functions which can retain some theoretical guarantees via injected randomness (or diversity) in hidden weights, while at the same time have the ability to adapt to the data at hand. More concretely, semi-random features are defined as

$$\phi_s(x; \mathbf{r}, \mathbf{w}) = \sigma_s(\mathbf{x}^\top \mathbf{r}) \left(\mathbf{x}^\top \mathbf{w}\right), \tag{7.1}$$

where $\mathbf{x} = (1, x^\top)^\top$ is assumed to be in $\mathbb{R}^{1+d}$, $\mathbf{r} = (r_0, r^\top)^\top$ is sampled randomly, and $\mathbf{w} = (w_0, w^\top)^\top$ is adjustable weights to be learned from data (hence, it is "semi-random"). Furthermore, the family of functions σ_s for $s \in \{0, 1, 2, \dots\}$ is defined as $\sigma_s(z) = (z)^s H(z)$, where H is Heaviside step function ($H(z) = 1$ for $z > 0$ and 0 otherwise). For instance, σ_0 is simply Heaviside step function, σ_1 is ramp function, and so on. We call the corresponding semi-random features with $s = 0$ "linear semi-random features (LSR)" and with $s = 1$ "squared semi-random features (SSR)". An illustration of example semi-random features can be found in Figure D.1 in Appendix D.1. Note that if we set $s = 0$ and $r := w$, the induced deterministic feature $\phi_{\text{deterministic}}(x; \mathbf{r}, \mathbf{w}) = \sigma_0(\mathbf{x}^\top \mathbf{w}) \left(\mathbf{x}^\top \mathbf{w}\right)$ is equivalent to that of rectified linear unit (ReLU) used in deep learning literature. Unlike dropout, which uses a data independent random switching mechanism (during training), the random switching in semi-random feature depends on the input data x (during both training and testing).

Intuitively, models with semi-random features have more expressive power than those with random features because of the learnable unit parameter $\mathbf{w}$. Yet, these models are less flexible compared to neural networks, since the parameters in $\sigma_s(\mathbf{x}^\top \mathbf{r})$ is sampled randomly. Further discussion on such comparison is deferred to Section 7.7.

In practice, we can typically assume that $x \in \Omega \subseteq \mathbb{R}^d$ with some sufficiently large compact subspace Ω. Thus, given such a Ω, we use the following sampling procedure in order for our method to be efficiently executable in practice: r is sampled uniformly

from a d dimensional unit sphere $\mathbb{S}^{d-1}$, and r_0 is sampled such that the probability measure on any open ball in $\mathbb{R}$ with its center in $[-\text{radius}(\Omega), \text{radius}(\Omega)]$ is nonzero.[1] For example, uniform distribution that covers $[-\text{radius}(\Omega), \text{radius}(\Omega)]$ or normal distribution with a nonzero finite variance suffices the above requirement.

By using semi-random features, we will construct one hidden layer model in Section 7.4, and multilayer model in Section 7.5.

7.4 One Hidden Layer Model

With semi-random features ϕ_s in equation (7.1), we define one hidden layer model for nonlinear function as

$$\hat{f}_n^s(x; w) = \sum_{k=1}^{n} \phi_s(x; \mathbf{r}_k, \mathbf{w}_k^{(1)}) w_k^{(2)}, \tag{7.2}$$

where $\mathbf{r}_k$ is sampled randomly for $k \in \{2, 3, \cdots, n\}$ as described in Section 7.3, and $\mathbf{r}_1$ is fixed to be the first element of the standard basis as $\mathbf{r}_1 = \mathbf{e}_1 = (1, 0, 0, \cdots 0)^\top$ (to compactly represent a constant term in x). We can think of this model as one hidden layer model by considering $\phi_s(x; \mathbf{r}, \mathbf{w}_k^{(1)})$ as the output of k-th unit of the hidden layer, and $\mathbf{w}_k^{(1)}$ as adjustable parameters associated with this hidden layer unit. This way of understanding the model will become helpful when we generalize it to a multilayer model in Section 7.5. Note that $\hat{f}_n^s(x; w)$ is a nonlinear function of x. When it is clear, by the notation w, we denote all adjustable parameters in the entire model.

In matrix notation, the model in (7.2) can be rewritten as

$$\hat{f}_n^s(x; w) = \left(\sigma_s(\mathbf{x}^\top \mathbf{R}) \odot (\mathbf{x}^\top \mathbf{W}^{(1)})\right) W^{(2)}, \tag{7.3}$$

[1] The radius of a compact subspace Ω of $\mathbb{R}^d$ is defined as $\text{radius}(\Omega) = \sup_{x \in \Omega} \|x\|_2$ and a open ball is defined as $B(\bar{z}, \delta) = \{z : \|z - \bar{z}\| < \delta\}$ where $\bar{z}$ is the center of the ball.

where

$$\mathbf{W}^{(1)} = (\mathbf{w}_1^{(1)}, \mathbf{w}_2^{(1)}, \ldots, \mathbf{w}_n^{(1)}) \in \mathbb{R}^{(d+1)\times n},$$

$$\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n) \in \mathbb{R}^{(d+1)\times n}, \text{and}$$

$$\mathbf{W}^{(2)} = (w_1^{(2)}, \ldots, w_n^{(2)})^\top \in \mathbb{R}^{n+1}.$$

Here, $(M_1 \odot M_2)$ represents a Hadamard product of two matrices M_1 and M_2. Furthermore $\sigma_s(M)_{ij} = \sigma_s(M_{ij})$, given a matrix M of any size (with overloads of the symbol σ_s).

In the following subsections, we present our theoretical results for one hidden layer model. Their complete proofs are deferred to the appendix.

7.4.1 Universal Approximation Ability

We show that our model class has universal approximation ability. Given a finite s, our model class is defined as

$$\mathcal{F}_n^s = \{x \mapsto \hat{f}_n^s(x; w) \,|\, w \in \mathbb{R}^{d_w}\},$$

where $d_w = (d+1)n + n$ is the number of adjustable parameters. Let $L^2(\Omega)$ be the space of square integrable functions on a compact set $\Omega \subseteq \mathbb{R}^d$. Then Theorem 32 states that we can approximate any $f \in L^2(\Omega)$ arbitrarily well as we increase the number of units n. We discuss the importance of the bias term r_0 to obtain the universal approximation power in Appendix D.2.

Theorem 32 (Universal approximation) *Let s be any fixed finite integer and let $\Omega \neq \{0\}$ be any fixed nonempty compact subset of $\mathbb{R}^d$. Then, for any $f \in L^2(\Omega)$, with probability one,*

$$\lim_{n\to\infty} \inf_{\hat{f}\in\mathcal{F}_n^s} \|f - \hat{f}\|_{L^2(\Omega)} = 0.$$

Proof Sketch. Based on a known result [137, Proposition 1], we prove the universality of σ_s with parameters $(r_0, r) \in \mathbb{R}^d \times \mathbb{R}$. Then, we show that the universality is preserved with

$(r_0, r) \in \mathbb{S}^{d-1} \times [-\text{radius}(\Omega), \text{radius}(\Omega)]$ (with constant functions that are included by the definition of $\mathbf{r}_1$). This means that if we try all *uncountably* many (r_0, r) in the set, we get the desired convergence. We translate this statement with a *uncountable* set to *countable* samples via continuity and probability measure; that is, if we try a sample $\mathbf{r}$ close enough to a desired $\mathbf{r}^*$, then the error gets arbitrarily small (continuity). In addition, the probability of sampling $\mathbf{r}$ close enough to a desired $\mathbf{r}^*$ reaches one as $n \to \infty$ (probability measure).

□

7.4.2 Optimization Theory

As we have confirmed universal approximation ability of our model class $\mathcal{F}_n^s$ in the previous section, we now want to find a good $\hat{f} \in \mathcal{F}_n^s$ via empirical loss minimization. More specifically, given a dataset $\{(x_i, y_i)\}_{i=1}^m$, we will consider the following optimization problem:

$$\underset{w \in \mathbb{R}^{d_w}}{\text{minimize}} \ \mathcal{L}(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \hat{f}_n^s(x_i; w) \right)^2.$$

Let $Y = (y_1, y_2, \ldots, y_m)^\top \in \mathbb{R}^m$ and $\hat{Y} = (f_n^s(x_1; w), f_n^s(x_2; w), \ldots, f_n^s(x_m; w))^\top \in \mathbb{R}^m$. Given a matrix M, let $\mathbf{P}_{\text{col}(M)}$ and $\mathbf{P}_{\text{null}(M)}$ be the projection matrices onto the column space and null space of M.

Our optimization problem turns out to be characterized by the following m by nd matrix:

$$D = \begin{bmatrix} \sigma_s(\mathbf{x}_1^\top \mathbf{r}_1)\mathbf{x}_1^\top & \cdots & \sigma_s(\mathbf{x}_1^\top \mathbf{r}_n)\mathbf{x}_1^\top \\ \vdots & \ddots & \vdots \\ \sigma_s(\mathbf{x}_m^\top \mathbf{r}_1)\mathbf{x}_m^\top & \cdots & \sigma_s(\mathbf{x}_m^\top \mathbf{r}_n)\mathbf{x}_m^\top \end{bmatrix}, \tag{7.4}$$

where $\sigma_s(\mathbf{x}_i^\top \mathbf{r}_j)\mathbf{x}_i^\top$ is a $1 \times d$ block at (i, j)-th block entry. That is, at any global minimum, we have $\mathcal{L}(w) = \frac{1}{2m} \|\mathbf{P}_{\ker(D^T)} Y\|^2$ and $\hat{Y} = \mathbf{P}_{\text{col}(D)} Y$. Moreover, we can achieve a global

minimum in polynomial time in d_w based on the following theorem.

Theorem 33 (No bad local minima and few bad critical points) *For any s and any n, the optimization problem of $\mathcal{L}(w)$ has the following properties:*

(i) it is non-convex (if $D \neq 0$)[2],

(ii) every local minimum is a global minimum,

(iii) if $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$, every critical point is a global minimum, and

(iv) at any global minimum, $\mathcal{L}(w) = \frac{1}{2m}\|\mathbf{P}_{\mathrm{null}(D^T)}Y\|^2$ and $\hat{Y} = \mathbf{P}_{\mathrm{col}(D)}Y$.

Proof Sketch. We first prove (iii) and (iv), by showing that if $\hat{Y} = \mathbf{P}_{\mathrm{col}(D)}Y$, it is a global minimum, and that $\hat{Y} = \mathbf{P}_{\mathrm{col}(D)}Y$ is achieved by any critical point if $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$. These two statements together imply (iii) and (iv). We then prove (ii) by showing that if a point is a local minimum with $w_k^{(2)} = 0$ for some $k \in \{1, 2, \ldots, n\}$, then the model output is $\hat{Y} = \mathbf{P}_{\mathrm{col}(D)}Y$, which is a global minimum. We prove (i) by showing that Hessian is not positive semidefinite at all w (if $D \neq 0$). □

Theorem 33 (optimization) together with Theorem 32 (universality) suggests that not only does our model class contain an arbitrarily good function (as n increases), but also we can find the best function in the model class given a dataset. In the context of understanding the loss surface of neural networks (e.g., see [114] for the information of its recent literature), Theorem 33 implies that all potential problems in the loss surface are due to the inclusion of $\mathbf{r}$ as an optimization variable. Optimizing over $\mathbf{r}$ as well as w increases the expressive power of the model class, but it would potentially make the optimization problem challenging. Semi-random features with fixed random $\mathbf{r}$ avoid this problem.

[2]In the case of $D = 0$, $\mathcal{L}(w)$ is convex in a trivial way; our model class only contains a single constant function $x \mapsto 0$.

7.4.3 Generalization Guarantee

In the previous sections, we have shown that our model class has universal approximation ability and that we can learn the best model given a finite dataset. A major remaining question is about the generalization property; how well can a learned model generalize to unseen new observations? Theorem 34 bounds the generalization error; the difference between expected risk, $\frac{1}{2}\mathbb{E}_x(f(x) - \hat{f}(x; w^*))^2$, and empirical risk, $\mathcal{L}(w)$.

Theorem 34 (Generalization bound for shallow model) *Let $s \geq 0$ and $n > 0$ be fixed. Let $\|\mathbf{x}\|_2 \leq C_X$ for all x. Consider any model class $\mathcal{F}_n^s$ with $\|\sigma_s(\mathbf{x}^\top \mathbf{R})\|_2 \leq C_\sigma$, $\|\mathbf{W}^{(1)}\|_2 \leq C_{W^{(1)}}$ and $\left\|W^{(2)}\right\|_2 \leq C_{W^{(2)}}$. Then, with probability at least $1 - \delta$, for any $\hat{f} \in \mathcal{F}_n^s$,*

$$\begin{aligned}&\frac{1}{2}\mathbb{E}_x(f(x) - \hat{f}(x; w))^2 - \mathcal{L}(w) \\ &\leq (C_Y^2 + C_{\hat{Y}}^2)\sqrt{\frac{\log \frac{1}{\delta}}{2m}} + 2(C_Y + C_{\hat{Y}})\frac{C_{\hat{Y}}}{\sqrt{m}},\end{aligned}$$

where $C_{\hat{Y}} = C_X C_\sigma C_{W^{(1)}} C_{W^{(2)}}$.

Proof Sketch. The proof is based on a standard use of Rademacher complexity (e.g., see section 3 in [138] for a clear introduction of Rademacher complexity). By formulating our model in a matrix form, we directly compute empirical Rademacher complexity from its definition. □

Here, $\|\mathbf{W}^{(1)}\|_2$ represents operator norm, but we can also bound it by Frobenius norm as $\|\mathbf{W}^{(1)}\|_2 \leq \|\mathbf{W}^{(1)}\|_F$. By combining Theorem 33 (optimization) and Theorem 34 (generalization), we obtain the following remark.

Remark 1. *(Expected risk bound)* Let $C_{W^{(1)}}$ and $C_{W^{(2)}}$ be values such that the global minimal value $\mathcal{L}(w) = \frac{1}{2m}\|\mathbf{P}_{\ker(D^T)}Y\|^2$ is attainable in the model class (e.g., setting $C_{W^{(2)}} = 1$ and $C_{W^{(1)}} = \|(D^\top D)^\dagger D^\top Y\|$ suffices). Then, at any critical points with $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$ and any local minimum such that $\|\mathbf{W}^{(1)}\|_2 < C_{W^{(1)}}$

and $\left\|W^{(2)}\right\|_2 < C_{W^{(2)}}$, we have

$$\mathbb{E}_x(f(x) - \hat{f}(x; w^*))^2 \leq \frac{\|\mathbf{P}_{\text{null}(D^T)}Y\|^2}{m} + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{m}}\right), \tag{7.5}$$

with probability at least $1 - \delta$. Here, $O(\cdot)$ notation simply hides the constants in Theorem 34.

In the right-hand side of equation (7.5), the second term goes to zero as m increases, and the first term goes to zero as n or d increases (because $\text{null}(D^T)$ becomes a smaller and smaller space as n or d increases, eventually containing only 0). Hence, we can minimize the expected risk to zero. We can also rewrite equation (7.5) if we adopt a set of additional assumptions used in [136] as follows:

Remark 2. Assume (only in this remark) that x is drawn uniformly from a unit sphere, $w_k^{(2)} \in \{-1, 1\}$ for all $k \in \{1, 2, \cdots, n\}$, and n and d are sufficiently large (as specified in [136, Theorem 1]). Then, with high probability, equation (7.5) holds with the first term in the right-hand side being replaced by $c\|\frac{\partial \mathcal{L}}{\partial W^{(1)}}\|$ for some constant c. [136] proved this special case of the bound, additionally assuming the enough randomness of the non-randomized version of $\mathbf{r}$ in the hidden layer.

7.5 Multilayer Model

We generalize one hidden layer model to H hidden layer model by composing semi-random features in a nested fashion. More specifically, let n_l be the number of units, or width, in the l-th hidden layer for all $l = 1, 2, \ldots, H$. Then we will denote a model of fully-connected feedforward semi-random networks with H hidden layers by

$$\hat{f}^s_{n_1,\ldots,n_H}(x; w) = h_w^{(H)}\, W^{(H+1)}, \tag{7.6}$$

where for all $l \in \{2, 3, \cdots, H\}$,

$$h_w^{(l)}(x) = h_r^{(l)}(x) \odot (h_w^{(l-1)}(x)\, W^{(l)}) \text{ and}$$
$$h_r^{(l)}(x) = \sigma_s(h_r^{(l-1)}(x)\, R^{(l)})$$

is the output of the l-th *semi-random* hidden layer, and the output of the l-th *random* switching layer respectively. Here, $W^{(l)} = (w_1^{(l)}, w_2^{(l)}, \ldots, w_{n_l}^{(l)}) \in \mathbb{R}^{n_{l-1} \times n_l}$ and $R^{(l)} = (r_1^{(l)}, r_2^{(l)}, \ldots, r_{n_l}^{(l)}) \in \mathbb{R}^{n_{l-1} \times n_l}$. Similarly to one hidden layer model, $r_k^{(l)}$ is sampled randomly for $k \in \{2, 3, \ldots, n_l\}$ but $r_1^{(l)}$ is fixed to be $\mathbf{e}_1 = (1, 0, 0, \ldots, 0)^\top$ (to compactly write the effect of constant terms in x). The output of the first hidden layer is the same as that of one hidden layer model:

$$h_w^{(1)}(x) = h_r^{(1)}(x) \odot (\mathbf{x}\mathbf{W}^{(1)}) \text{ and } h_r^{(1)}(x) = \sigma_s(\mathbf{x}\mathbf{R}^{(1)}),$$

where the boldface notation emphasizes that we require the bias terms at least in the first layer. In other words, we keep randomly updating the random switching layer $h_r^{(l)}(x)$, and couple it with a linearly adjustable hidden layer $h_w^{(l)}(x)W^{(l)}$ to obtain the next semi-random hidden layer $h_w^{(l+1)}(x)$.

Convolutional semi-random feedforward neural networks can be defined in the same way as in equation (7.6) with vector-matrix multiplication being replaced by c dimensional convolution (for some number c). In our experiments, we will test both convolutional semi-random networks as well as fully-connected versions.

In the following sections, we present our analysis for multilayer *fully-connected* semi-random networks. The complete proofs are deferred to the appendix.

7.5.1 Tensorial Structure

Since the output of our network is the sum of the outputs of all the paths in the network, we observe the following interesting structure:

$$\hat{f}^s_{n_1,\ldots,n_H}(x) = \sum_{k_0=0}^{d} \sum_{k_1=1}^{n_1} \cdots \sum_{k_H=1}^{n_H} [[\sigma]]_{k_1,\ldots,k_H}(x) \mathbf{x}_{k_0} [[w]]_{k_0,k_1,\ldots,k_H},$$

where $[[\sigma]]_{k_1,\ldots,k_H}(x) = \sigma_s(\mathbf{x}^\top \mathbf{r}^{(1)}_{k_1}) \prod_{l=2}^{H} \sigma_s(h_r^{(l-1)}(x)\, r^{(l)}_{k_l})$, and $[[w]]_{k_0,k_1,\ldots,k_H} = \left(\prod_{l=1}^{H} w^{(l)}_{k_{l-1}k_l}\right) w^{(H+1)}_{k_H}$. This means that the function is the weighted combination of $(d+1) \times n_1 \times n_2 \ldots n_H$ nonlinear basis functions, which is exponential in the number of layers H. Alternatively, the function can also be viewed as the inner product between two tensors $[[\sigma]]_{k_1,\ldots,k_H}(x)\mathbf{x}_{k_0}$ and $[[w]]_{k_0,k_1,\ldots,k_H}$, which is nonlinear in input x, but *linear* in the parameter tensor $[[w]]_{k_0,k_1,\ldots,k_H}$. We note that the parameter tensor is not arbitrary but highly structured: it is composed using a collection of matrices with $d_w = (d+1)n_1 + n_H + \sum_{l=2}^{H} n_{l-1}n_l$ number of adjustable parameters. Such special structure allows the function to generate an exponential number of basis functions yet keep the parameterization compact.

7.5.2 Benefit of Depth

We first confirm that our multilayer model class

$$\mathcal{F}^s_{n_1,\ldots,n_H} = \{x \mapsto \hat{f}^s_{n_1,\ldots,n_H}(x; w) | w \in \mathbb{R}^{d_w}\}$$

preserves universal approximation ability.

Corollary 35 (Universal approximation with deep model) *Let s be any fixed finite integer and let $\Omega \neq \{0\}$ be any fixed nonempty compact subset of $\mathbb{R}^d$. Then, for any $f \in L^2(\Omega)$, with probability one,*

$$\lim_{n_1,\ldots,n_H \to \infty} \inf_{\hat{f} \in \mathcal{F}^s_{n_1,\ldots,n_H}} \|f - \hat{f}\|_{L^2(\Omega)} = 0.$$

We now know that both of one hidden layer models and deeper models have universal approximation ability. Then, a natural question arises: how can depth benefit us? To answer the question, note that as illustrated in Section 7.5.1, H hidden layer model only needs $O(nH)$ number of parameters (by setting $n = n_1, n_2, \cdots, n_H$) to create around $n^H d$ paths, whereas one hidden layer model requires $O(n^H)$ number of parameters to do so. Intuitively, because of this, the expressive power would grow exponentially in depth H, if those exponential paths are not redundant to each other. The redundancy among the paths would be minimized via randomness in the switching and exponentially many combinations of nonlinearities σ_s.

We formalize this intuition by considering concrete degrees of approximation powers for our models. To do so, we adopt a type of a degree of "smoothness" on the target functions from a previous work [109]. Consider Fourier representation of a target function as $f(x) = \int_{\omega \in \mathbb{R}^d} \tilde{f}(\omega) e^{i\omega^\top x}$. Define a class of smooth functions Γ_C:

$$\Gamma_C = \left\{ x \mapsto f(x) : \int_{\omega \in \mathbb{R}^d} \|\omega\|_2 |\tilde{f}(\omega)| \leq C \right\}.$$

Any $f \in \Gamma_C$ with finite C is continuously differentiable in $\mathbb{R}^d$, and the gradient of f can be written as $\nabla_x f(x) = \int_{\omega \in \mathbb{R}^d} i\omega \tilde{f}(\omega) e^{i\omega^\top x}$. Thus, via Plancherel theorem, we can view C as the bound on $\|\nabla_x f(x)\|_{L(\mathbb{R}^d)}$. See the previous work [109] for a more detailed discussion on the properties of this function class Γ_C. Theorem 36 states that a lower bound on the approximation power gets better exponentially in depth H.

Theorem 36 (Lower bound on universal approximation power) *Let $\Omega = [0,1]^d$. For every fixed finite integer s, for any depth $H \geq 0$, and for any set of nonzero widths $\{n_1, n_2, \ldots, n_H\}$,*

$$\sup_{f \in \Gamma_C} \inf_{\hat{f} \in \mathcal{F}^s_{n_1 \cdots n_H}} \|f - \hat{f}\|_{L^2(\Omega)} \geq \frac{\kappa C}{d^2} \left(\prod_{l=1}^{H} n_l \right)^{-1/d},$$

where $\kappa \geq (8\pi e^{(\pi-1)})^{-1}$ is a constant.

Proof Sketch. We formalize the intuition discussed above. That is, by expressing our model as a sum of the paths as in Section 7.5.1, we observe that our model class is included in the span of functions associated with the paths, the number of which grows exponentially in depth H. Since a span of functions of a fixed number cannot approximate any $f \in \Gamma_C$ arbitrarily well, based on a known result [109, Theorem 6], we obtain the lower bound. □

By setting $n = n_1 = n_2 = \cdots = n_H$, the lower bound in Theorem 36 becomes:

$$\sup_{f \in \Gamma_C} \inf_{\hat{f} \in \mathcal{F}^s_{n_1 \cdots n_H}} \|f - \hat{f}\|_{L^2(\Omega)} \geq \frac{\kappa C}{d^2} n^{-H/d},$$

where we can easily see the benefit of the depth H. However, note that the lower bound is not intended to be tight here (indeed, it is proven by relaxation). Thus, we cannot make a formal comparison between H hidden layer model and one hidden layer model based on Theorem 36 alone. Our hope here is to provide a formal statement to aid intuitions. To help our intuitions further, we discuss about an upper bound on universal approximation power for multilayer model in Appendix D.5.

7.5.3 Optimization Theory

Similarly to one hidden layer case, we consider the following optimization problem:

$$\underset{w \in \mathbb{R}^{d_w}}{\text{minimize}}\, \mathcal{L}^{(H)}(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - f^s_{n_1,\ldots,n_H}(x; w)\right)^2.$$

Compared to one hidden layer case, our theoretical understanding of multilayer model is rather preliminary. Here, given a function $g(w_1, w_2, \ldots, w_n)$, we say that $\bar{w} = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n)$ is a global minimum of g with respect to w_1 if $\bar{w}_1$ is a global minimum of $\tilde{g}(w_1) = g(w_1, \bar{w}_2, \ldots, \bar{w}_n)$.

Corollary 37 (No bad local minima and few bad critical points w.r.t. last two layers) *For any s, any depth $H \geq 1$, and any set of nonzero widths $\{n_1, n_2, \ldots, n_H\}$, the optimization problem of $\mathcal{L}^{(H)}(w)$ has the following property:*

(i) every local minimum is a global minimum with respect to $(W^{(H)}, W^{(H+1)})$, and

(ii) if $w_k^{(H+1)} \neq 0$ for all $k \in \{1, 2, \ldots, n_H\}$, every critical point is a global minimum with respect to $(W^{(H)}, W^{(H+1)})$.

Future work is still required to investigate the theoretical nature of the optimization problem with respect to all parameters. Some hardness results of a standard neural network optimization come from the difficulty of learning activation pattern via optimization of the variable $\mathbf{r}$ [139]. In this sense, our optimization problem is somewhat easier, and it would be interesting to see if we can establish meaningful optimization theory for semi-random model as a first step to establish the theory for neural networks in general.

7.5.4 Generalization Guarantee

The following corollary bounds the generalization error. In the statement of the corollary, we can easily see that the generalization error goes to zero as m increases (as long as the relevant norms are bounded). Hence, we can achieve generalization.

Corollary 38 (Generalization bound for deep model) *Let $s \geq 0$ and $H \geq 1$ be fixed. Let $n_l > 0$ be fixed for $l = 1, 2, \ldots, H$. Let $\|\mathbf{x}\|_2 \leq C_X$ for all x. Consider the model class $\mathcal{F}^s_{n_1,\ldots,n_H}$ with $\left\|W^{(l)}\right\|_2 \leq C_{W^{(l)}}$ and $\|h_r^{(l)}(x)\|_2 \leq C_{\sigma^{(l)}}$ for $l = 1, 2, \cdots, H$. Then, with probability at least $1 - \delta$, for any $\hat{f} \in \mathcal{F}^s_{n_1,\ldots,n_H}$,*

$$\frac{1}{2}\mathbb{E}_x(f(x) - \hat{f}(x; w^*))^2 - \mathcal{L}^{(H)}(w) \leq (C_Y^2 + C_{\hat{Y}}^2)\sqrt{\frac{\log \frac{1}{\delta}}{2m}} + 2(C_Y + C_{\hat{Y}})\frac{C_{\hat{Y}}}{\sqrt{m}},$$

where $C_{\hat{Y}} = C_X C_{W^{(H+1)}} \prod_{l=1}^{H} C_{\sigma^{(l)}} C_{W^{(l)}}$.

In Corollary 38, a generalization bound gets worse in depth H if the operator norm of each layer's weights is larger than one. By controlling the norms, we can control the generalization bound.

7.6 Experiments

We compare semi-random features with random features (RF) and neural networks with ReLU on both UCI datasets and image classification benchmarks. We will study two variants of semi-random features for $s = 0$ (LSR: linear semi-random features) and $s = 1$ (SSR: squared semi-random features) in $\sigma_s(\cdot)$ from equation (7.1). Additional experimental details are presented in Appendix D.6. The source code of the proposed method is publicly available at: http://github.com/zixu1986/semi-random.

7.6.1 Simple Test Function

We first tested the methods with a simple sine function, $f(x) = \sin(x)$, where we can easily understand what is happening. Figure 7.1 shows the test errors with one standard deviations. As we can see, semi-random network (LSR) performed the best. The problem of ReLU became clear once we visualized the function learned at each iteration: ReLU network had a difficulty to diversify activation units to mimic the frequent oscillations of the sine function (i.e., it took long time to diversely allocate the breaking points of its piecewise linear function). The visualizations of learned functions at each iteration for each method are presented in Appendix D.6. On average, they took 54.39 (ReLU), 43.04 (random), 45.44 (semi-random) seconds. Their training errors are presented in Appendix D.6.

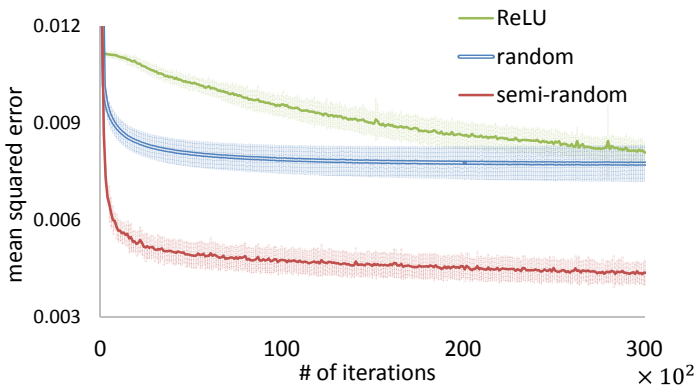


Figure 7.1: Test error for a simple test function.

7.6.2 UCI datasets

We have comparisons on six large UCI datasets.[3] The network architecture used on this dataset is multi-layer networks with $l = [1, 2, 4]$ hidden layers and $k = [1, 2, 4, 8, 16] \times d$ hidden units per layer where d is the input data dimension.

Comparison of best performance. In Table 7.1, we listed the best performance among different architectures for each method. On most datasets, ReLU has the lowest error while random features have the highest error. In comparison, semi-random features achieve significant lower errors than random features and the errors are close to that of ReLU.

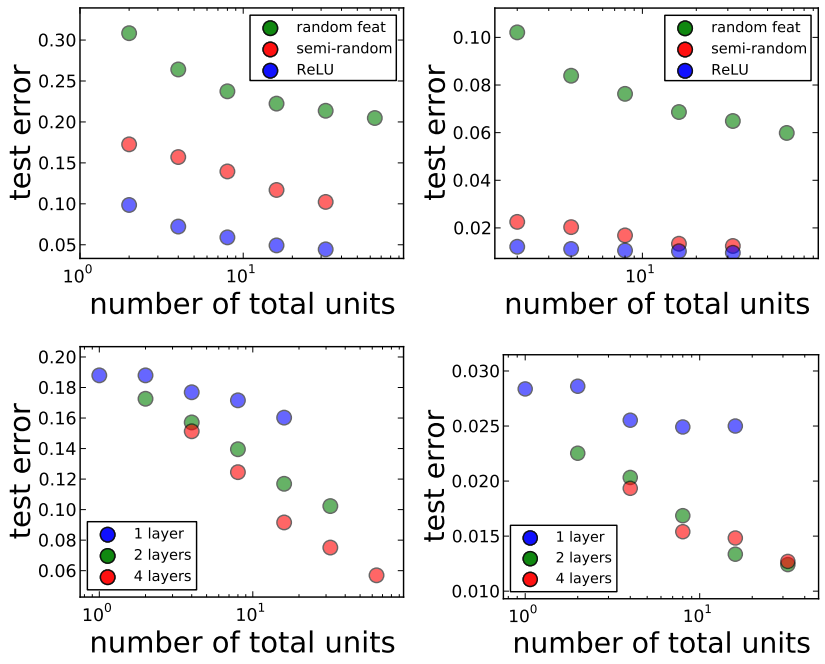
Matching the performance of ReLU. The top row of Figure 7.2 demonstrates how many more units are required for random and semi-random features to reach the test errors of neural networks with ReLU. First, all three methods enjoy lower test errors by increasing the number of hidden units. Second, semi-random units can achieve comparable performance to ReLU with slightly more units, around 2 to 4 times in Webspam dataset. In comparison, random features require many more units, more than 16 times. These experiments clearly show the benefit of adaptivity in semi-random features.

Depth vs width. The bottom row of Figure 7.2 explores the benefit of depth. Here, "l-layer" indicates l *hidden* layer model. To grow the number of total units, we can either use more layers or more units per layer. Experiment results suggest that we can gain more in

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

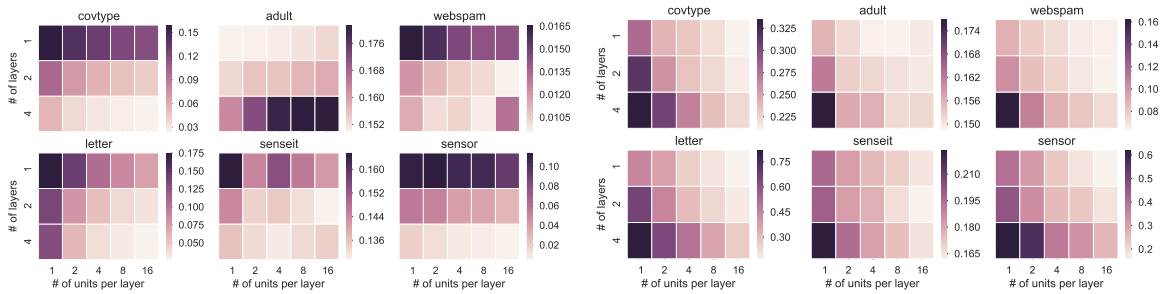
Table 7.1: Performance comparison on UCI datasets. RF for random features, LSR and SSR for linear ($s = 0$) and squared ($s = 1$) semi-random features respectively. m_{tr}: number of training data points; m_{te}: number of test data points; d dimension of the data

Dataset	method	err (%)	Dataset	method	err (%)
covtype	ReLU	2.3	adult	ReLU	15.0
$m_{\text{tr}} = 522,910$	RF	20.2	$m_{\text{tr}} = 32,561$	RF	14.9
$m_{\text{te}} = 58,102$	LSR	5.7	$m_{\text{te}} = 16,281$	LSR	14.8
$d = 54$	SSR	14.4	$d = 123$	SSR	14.9
webspam	ReLU	1.0	senseit	ReLU	12.6
$m_{\text{tr}} = 280,000$	RF	6.0	$m_{\text{tr}} = 78,823$	RF	16.0
$m_{\text{te}} = 70,000$	LSR	1.1	$m_{\text{te}} = 19,705$	LSR	13.9
$d = 123$	SSR	2.2	$d = 100$	SSR	13.3
letter	ReLU	2.7	sensor	ReLU	0.4
$m_{\text{tr}} = 15,000$	RF	14.9	$m_{\text{tr}} = 48,509$	RF	13.4
$m_{\text{tr}} = 5,000$	LSR	6.5	$m_{\text{tr}} = 10,000$	LSR	1.4
$d = 16$	SSR	5.6	$d = 48$	SSR	5.7



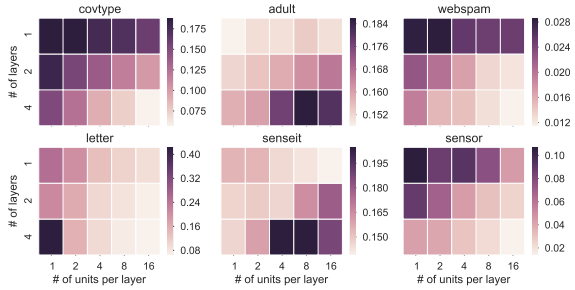
(a) Covtype dataset (b) Webspam dataset

Figure 7.2: Top row: Linear semi-random features match the performance of ReLU for two hidden layer networks in two datasets. Bottom row: Depth vs width of linear semi-random features. Both plots show performance of semi-random units. Even the total number of units is the same, deeper models achieve lower test error.

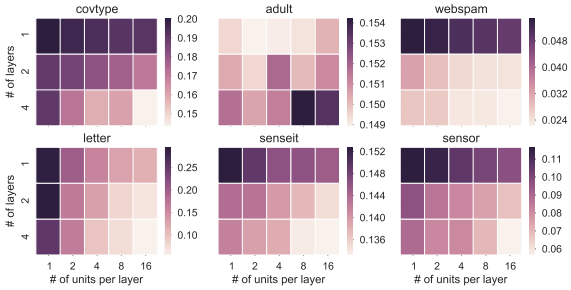


(a) ReLU

(b) random features



(c) linear semi-random features



(d) squared semi-random features

Figure 7.3: Detailed experiment results for all types of neurons and on all datasets. The heat map for each dataset shows how the test error changes w.r.t. the number of layers and number of units per layer.

performance by going deeper. The ability to benefit from deeper architecture is an important feature that is not possessed by random features.

The details on how the test error changes w.r.t. the number of layers and number of units per layer are shown in Figure 7.3. As we can see, on most datasets, more layers and more units lead to smaller test errors. However, the adult dataset is more noisy and it is easier to overfit. All types of neurons perform relatively the same on this dataset, and more parameters actually lead to worse results. Furthermore, the squared semi-random features have very similar error pattern to neural network with ReLU.

7.6.3 Image classification benchmarks

We have also compared different methods on three image classification benchmark datasets. Here we use publicly available and well-tuned neural network architectures from tensorflow for the experiments. We simply replace ReLU by random and semi-random units

respectively. The results are summarized in Table 7.2.

MNIST dataset. MNIST is a popular dataset for recognizing handwritten digits. It contains 28×28 grey images, 60,000 for training and 10,000 for test. We use a convolution neural network consisting of two convolution layers, with 5×5 filters and the number of channels is 32 and 64, respectively. Each convolution is followed by a max-pooling layer, then finally a fully-connected layer of 512 units with 0.5 dropout.

ReLU units achieve the best test error of 0.70%. Increasing the number of units for semi-random leads to better performance. At four times the size of the original network, semi-random feature can achieve very close errors of 0.71%. In contrast, even when increasing the number of units to 16 times more, random features still cannot reach below 1%.

CIFAR10 dataset. CIFAR 10 contains internet images and consists of 50,000 32×32 color images for training and 10,000 images for test. We use a convolutional neural network architecture with two convolution layers, each with 64 5×5 filters and followed by max-pooling. The fully-connected layers contain 384 and 192 units.

For this particular network architecture, ReLU has the best performance. By using two times more units, semi-random features are able to achieve similar performance. However, the performance of random features lags behind by a huge margin.

SVHN dataset. The Street View House Numbers (SVHN) dataset contains house digits collected by Google Street View. We use the 32×32 color images version and only predict the digits in the middle of the image. For training, we combined the training set and the extra set to get a dataset with 604,388 images. We use the same architecture as in the CIFAR10 experiments. ReLU has the lowest error of 3.9% while semi-random units achieve close error of 6.4%. Random features suffer from huge errors.

Table 7.2: Test error (in %) of different methods on three image classification benchmark datasets. $2\times$, $4\times$ and $16\times$ mean the number of units used is 2 times, 4 times and 16 times of that used in neural network with ReLU respectively.

neuron type	MNIST	CIFAR10	SVHN
ReLU	0.70	16.3	3.9
RF	8.80	59.2	73.9
RF $2\times$	5.71	55.8	70.5
RF $4\times$	4.10	49.8	58.4
RF $16\times$	2.69	40.7	37.1
LSR	0.97	21.4	7.6
LSR $2\times$	0.78	17.4	6.9
LSR $4\times$	0.71	18.7	6.4
SSR	1.14	26.3	10.0
SSR $2\times$	0.86	22.9	12.8
SSR $4\times$	0.78	20.9	8.1

7.7 Better than Random Feature?

The experimental results verified our intuition that semi-random feature can outperform random feature with fewer number of unites due to its learnable weights. We can also strengthen this intuition via the following theoretical insights. Let $f \in \mathcal{F}_{\text{random},n_1\cdots n_H}$ be a function that is a composition of any fully-random features with depth H where the adjustable weights are only in the last layer.

The following corollary states that a model class of any fully-random features has a approximation power exponentially bad in the dimensionality of x in the worst case.

Corollary 39 (Lower bound on approximation power for fully-random feature) *Let* $\Omega = [0,1]^d$. *For any depth* $H \geq 0$, *and for any set of nonzero widths* $\{n_1, n_2, \ldots, n_H\}$,

$$\sup_{f\in\Gamma_C} \inf_{\hat{f}\in\mathcal{F}_{\text{random},n_1\cdots n_H}} \|f - \hat{f}\|_{L^2(\Omega)} \geq \frac{\kappa C}{d^2}(n_H)^{-1/d},$$

where $\kappa \geq (8\pi e^{(\pi-1)})^{-1}$ *is a constant.*

Corollary 39 (lower bound for fully-random feature) together with Theorem 36 (lower

bound for semi-random feature) reflects our intuition that semi-random feature model can potentially get exponential advantage over random feature by learning hidden layer's weights. Again, because the lower bound may not be tight, this is intended only to aid our intuition.

We can also compare upper bounds on their approximation errors with an additional assumption. Assume that we can represent a target function f using some basis as

$$f(x) = \int_{r \in \mathbb{S}^{d-1}, \|w\| \leq C_W} \sigma(r^\top x)(w^\top x)\, p(r, w).$$

Then, we can obtain the following results.

- If we have access to the true distribution $p(r, w)$, $f(x)$ can be approximated as a finite sample average, obtaining approximation error of $O(\frac{1}{\sqrt{n}})$.
- Without knowing the true distribution $p(r, w)$, a purely random feature approximation of $f(x)$ incurs a large approximation error of $O(\frac{c}{q_0 \cdot q_1 \cdot \sqrt{n}})$. Here, q_0 is inverse of the hyper-surface area of a unit hypersphere and q_1 is the inverse of the volume of a ball of radius C_W.
- Without knowing the true distribution $p(r, w)$, semi-random feature approach with one hidden layer model can obtain a smaller approximation error of $O(\frac{35c}{\sqrt{n}})$.

A detailed derivation of this result is presented in Appendix D.8.

7.8 Summary

In this chapter, we proposed the method of semi-random features. For one hidden layer model, we proved that our model class contains an arbitrarily good function as the width increases (universality), and we can find such a good function (optimization theory) that generalizes to unseen new data (generalization bound). For deep model, we proved universal approximation ability, a lower bound on approximation error, a partial optimization guarantee, and a generalization bound. Furthermore, we demonstrated the advantage of

semi-random features over fully-random features via empirical results and theoretical insights. We have discussed several open questions in relevant sections.

CHAPTER 8
CONCLUSION

Non-convex optimization problems are ubiquitous in machine learning. Non-convex formulations are necessary because they capture some important aspects of the problems and they can more efficiently represent a problem. General non-convex problems are hard but empirical success of simple algorithms in machine learning tasks suggest that there are more structures in these specific problems.

In this thesis, we propose efficient algorithms and provide novel analysis for some of the non-convex optimization problems in machine learning. We show that by leveraging the structures in the models, we can overcome the non-convexity and obtain algorithms that are efficient with provable guarantees. In particular, we tackle two important classes of non-convex problems: 1) latent variable models, and 2) deep neural networks.

For multi-view latent variable models, we propose a nonparametric spectral algorithm to estimate the parameters. Compared with the Expectation Maximization (EM) algorithm, our proposed algorithm is guaranteed to return the true parameter when the number of data points tend to infinity. In addition, the nonparametric nature of the algorithm enables it to model any arbitrary conditional distribution, and thus it outperforms alternatives that can only model discrete or parametric distributions.

The key computation of the nonparametric spectral method boils down to that of kernel principle component analysis (KPCA). In order to fully utilize the potential of nonparametric methods, we propose two scalable algorithms for the non-convex problem of KPCA. The first one is a doubly stochastic gradient algorithm that makes two stochastic approximations. The first approximation is to the expectation over dataset and the second is to the expectation over random features. By growing the model adaptively as more data arrive, the algorithm strikes a better balance between computation and statistics. The other scalable

algorithm is a distributed version of KPCA. It selects a small representative subset of data points according to the leverage scores to reduce computation. In order to reduce communication overhead, the algorithm estimates the leverage scores through random sketching. Overall, the algorithm achieves nearly optimal communication efficiency.

For deep neural network problems, we have provided novel analysis to show why the optimization landscape of one-hidden-layer neural network is especially nice. In particular, when the neural weights are diverse enough, then there are no spurious local optima. This partly explains why simple gradient descent works so well for neural networks in practice since a stationary point obtained by gradient descent is likely to be a global optimum. Inspired by the insight from the analysis, we have further proposed semi-random units which sits between a fully adjustable ReLU unit and a fully random unit. Semi-random features enjoy several nice theoretical properties and yet still retain most of the representation power of a fully adjustable unit. In experiment, it can achieve much lower error compared with random features and almost approach the performance of fully adjustable units by using slightly more units.

Non-convex optimization problems are increasingly more important given the phenomenal success of deep learning. To solve these problems, this thesis contributes both algorithmic and theoretical tools from the perspective of spectral methods and analysis of spectral properties. In latent variable models, spectral decomposition is the key computation to recover the parameters with guarantees. The scalable algorithms proposed in this thesis focus on solving large nonlinear eigen-problems. Moreover, the analysis of deep neural network centers around the spectral property of some special matrix.

The research presented in this thesis opens up some new directions and open questions for further investigation. First of all, we can adapt the same technique of analyzing one-hidden-layer neural network to understand the optimization landscape of deeper architectures. In this setting, the special matrix then consists of effects from both lower and upper layers. In order to isolate the complicated nonlinear effects that are difficult to analyze, we

can borrow the strategies from analyzing residual neural networks [140]. The significant term from the residual network architecture will be the identity transform and thus gives a lower bound on the complicated nonlinear effects. Second, a direct analysis of gradient descent is needed to understand the exact mechanisms that occur in practice. Although we have shown the optimization landscape exhibits nice regions, it requires more research to confirm that gradient descent indeed converges to these regions. Third, the tensorial view of the semi-random features representation allows us to approach the problem from a new perspective. Basically, the hierarchical structure generates an exponential number of bases with only polynomial number of parameters. What other structures can also efficiently generate an exponential number of bases with few parameters? More research along this line may open up new paradigms of learning.

Appendices

APPENDIX A

PROOFS AND ADDITIONAL EXPERIMENTS IN CHAPTER 2

A.1 Symmetrization

We presented the kernel algorithm for learning the multi-view latent variable model where the views have identical conditional distributions. In this section, we will extend it to the general case where the views are different. Without loss of generality, we will consider recover the operator $\mu_{X_3|h}$ for conditional distribution $\mathbb{P}(X_3|h)$. The same strategy applies to other views. The idea is to reduce the multi-view case to the identical-view case based on a method by [11].

Given the observations $\mathcal{D}_{X_1X_2X_3} = \{(x_1^i, x_2^i, x_3^i)\}_{i\in[m]}$ drawn *i.i.d.* from a multi-view latent variable model $\mathbb{P}(X_1, X_2, X_3)$, let the kernel matrix associated with X_1, X_2 and X_3 be K, L and G respectively and the corresponding feature map be ϕ, ψ and υ respectively. Furthermore, let the corresponding feature matrix be $\widetilde{\Phi} = (\phi(x_1^1), \ldots, \phi(x_1^m))$, $\widetilde{\Psi} = (\phi(x_2^1), \ldots, \phi(x_2^m))$ and $\widetilde{\Upsilon} = (\phi(x_3^1), \ldots, \phi(x_3^m))$. Then, we have the empirical estimation of the second/third-order embedding as

$$\widehat{\mathcal{C}}_{X_1X_2} = \frac{1}{m}\widetilde{\Phi}\widetilde{\Psi}^\top,\ \widehat{\mathcal{C}}_{X_3X_1} = \frac{1}{m}\widetilde{\Upsilon}\widetilde{\Phi}^\top,\ \widehat{\mathcal{C}}_{X_2X_3} = \frac{1}{m}\widetilde{\Psi}\widetilde{\Upsilon}^\top$$
$$\widehat{\mathcal{C}}_{X_1X_2X_3} := \frac{1}{m}\boldsymbol{I}_n \times_1 \widetilde{\Phi} \times_2 \widetilde{\Psi} \times_3 \widetilde{\Upsilon}$$

Find two arbitrary matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{k\times\infty}$, so that $\boldsymbol{A}\widehat{\mathcal{C}}_{X_1X_2}\boldsymbol{B}^\top$ is invertible. Theoretically, we could randomly select k columns from Φ and Ψ and set $\boldsymbol{A} = \Phi_k^\top, \boldsymbol{B} = \Psi_k^\top$. In practial, the first k leading eigenvector directions of respect *RKHS* works better. Then, we

have

$$\begin{aligned}
\widetilde{\mathcal{C}}_{X_1X_2} &= \frac{1}{m}\widetilde{\Phi}_k^\top\widetilde{\Phi}\widetilde{\Psi}^\top\widetilde{\Psi}_k = \frac{1}{m}K_{nk}^\top L_{nk} \\
\widetilde{\mathcal{C}}_{X_3X_1} &= \widehat{\mathcal{C}}_{X_3X_1}\widetilde{\Phi}_k = \frac{1}{m}\widetilde{\Upsilon}K_{nk} \\
\widetilde{\mathcal{C}}_{X_3X_2} &= \widehat{\mathcal{C}}_{X_3X_2}\widetilde{\Psi}_k = \frac{1}{m}\widetilde{\Upsilon}L_{nk} \\
\widetilde{\mathcal{C}}_{X_1X_2X_3} &= \widehat{\mathcal{C}}_{X_1X_2X_3} \times_1 \widetilde{\Phi}_k^\top \times_2 \widetilde{\Psi}_k^\top = \frac{1}{m}\boldsymbol{I}_n \times_1 K_{nk}^\top \times_2 L_{nk}^\top \times_3 \widetilde{\Upsilon}
\end{aligned}$$

Based on these matrices, we could reduce to a single view

$$\begin{aligned}
Pair_3 &= \widetilde{\mathcal{C}}_{X_3X_1}(\widetilde{\mathcal{C}}_{X_1X_2}^\top)^{-1}\widetilde{\mathcal{C}}_{X_3X_2} \\
&= \frac{1}{m}\widetilde{\Upsilon}K_{nk}(L_{nk}^\top K_{nk})^{-1}L_{nk}^\top\widetilde{\Upsilon}^\top = \frac{1}{m}\widetilde{\Upsilon}H\widetilde{\Upsilon}^\top
\end{aligned}$$

where $H = K_{nk}(\mathcal{L}_{nk}^\top K_{nk})^{-1}L_{nk}^\top$.

Assume the leading k eigenvectors ν_k lie in the span of the column of Υ, i.e., $\nu_k = \Upsilon\beta_k$ where $\beta_k \in \mathbb{R}^{m\times 1}$

$$\begin{aligned}
Pair_3\nu = \lambda\nu &\Rightarrow (Pair_3)^\top Pair_3\nu = \lambda^2\nu \\
&\Rightarrow \frac{1}{m^2}\widetilde{\Upsilon}H^\top\widetilde{\Upsilon}^\top\widetilde{\Upsilon}H\widetilde{\Upsilon}^\top\nu = \lambda^2\nu \\
&\Rightarrow \frac{1}{m^2}\widetilde{\Upsilon}H^\top GHG\boldsymbol{\beta} = \lambda^2\widetilde{\Upsilon}\boldsymbol{\beta} \\
&\Rightarrow \frac{1}{m^2}GH^\top GHG\beta = \lambda^2 G\beta
\end{aligned}$$

Then, we symmetrize and whiten the third-order embedding

$$Triple_3 = \frac{1}{m}\widetilde{\mathcal{C}}_{X_1X_2X_3} \times_1 [\widetilde{\mathcal{C}}_{X_3X_2}\widetilde{\mathcal{C}}_{X_1X_2}^{-1}] \times_2 [\widetilde{\mathcal{C}}_{X_3X_1}\widetilde{\mathcal{C}}_{X_2X_1}^{-1}] \tag{A.1}$$

Plug $\widetilde{\mathcal{C}}_{X_3X_2}\widetilde{\mathcal{C}}_{X_1X_2}^{-1} = \widetilde{\Upsilon}L_{nk}(K_{nk}^\top L_{nk})^{-1}$ and $\widetilde{\mathcal{C}}_{X_3X_1}\widetilde{\mathcal{C}}_{X_2X_1}^{-1} = \widetilde{\Upsilon}K_{nk}(L_{nk}^\top K_{nk})^{-1}$, we have

$$Triple_3 = \frac{1}{m}\boldsymbol{I}_n \times_1 \widetilde{\Upsilon} L_{nk}(K_{nk}^\top L_{nk})^{-1} K_{nk}^\top \\ \times_2 \widetilde{\Upsilon} K_{nk}(L_{nk}^\top K_{nk})^{-1} L_{nk}^\top \times_3 \Upsilon$$

We multiply each mode with $\Upsilon\beta\widehat{S}_k^{-\frac{1}{2}}$ to whitening the data and apply power method to decompose it

$$\begin{aligned}\widehat{\mathcal{T}} &= Triple_3 \times_1 \widehat{S}_k^{-\frac{1}{2}}\beta^\top\widetilde{\Upsilon}^\top \times_2 \widehat{S}_k^{-\frac{1}{2}}\beta^\top\widetilde{\Upsilon}^\top \times_3 \widehat{S}_k^{-\frac{1}{2}}\beta^\top\widetilde{\Upsilon}^\top \\ &= \frac{1}{m}\boldsymbol{I}_n \times_1 \widehat{S}_k^{-\frac{1}{2}}\beta^\top G\mathcal{L}_{nk}(K_{nk}^\top L_{nk})^{-1}K_{nk}^\top \times_2 \\ &\quad \widehat{S}_k^{-\frac{1}{2}}\beta^\top G K_{nk}(L_{nk}^\top K_{nk})^{-1}L_{nk}^\top \times_3 \widehat{S}_k^{-\frac{1}{2}}\beta^\top G\end{aligned}$$

A.2 Robust Tensor Power Method

We recap the robust tensor power method for finding the tensor eigen-pairs in Algorithm 8, analyzed in detail in [25] and [10]. The method computes the eigenvectors of a tensor through deflation, using a set of initialization vectors. Here, we employ random initialization vectors. This can be replaced with better initialization vectors, in certain settings, e.g. in the community model, the neighborhood vectors provide better initialization and lead to stronger guarantees [25]. Given the initialization vector, the method then runs a tensor power update, and runs for N iterations to obtain an eigenvector. The successive eigenvectors are obtained via deflation.

A.3 Proof of Theorem 2

A.3.1 Recap of Perturbation Bounds for the Tensor Power Method

We now recap the result of [25, Thm. 13] that establishes bounds on the eigen-estimates under good initialization vectors fciteor the above procedure. Let $\mathcal{T} = \sum_{i\in[k]} \lambda_i v_i$, where

Algorithm 8 $\{\lambda, M\} \leftarrow$TensorEigen$(\mathcal{T}, \{v_i\}_{i\in[k]}, N)$

Input: Tensor $\mathcal{T} \in \mathbb{R}^{k\times k\times k}$, set of k initialization vectors $\{v_i\}_{i\in[k]}$, number of iterations N.

Output: the estimated eigenvalue/eigenvector pairs $\{\lambda, M\}$, where $\lambda = (\lambda_1, \ldots, \lambda_k)^\top$ is the vector of eigenvalues and $M = (v_1, \ldots, v_k)$ is the matrix of eigenvectors.

for $i = 1$ to k **do**
  for $\tau = 1$ to k **do**
    $\theta_0 \leftarrow v_\tau$.
    for $t = 1$ to N **do**
      $\widetilde{\mathcal{T}} \leftarrow \mathcal{T}$.
      for $j = 1$ to $i - 1$ (when $i > 1$) **do**
        if $|\lambda_j \left\langle \theta_t^{(\tau)}, v_j \right\rangle| > \xi$ **then**
          $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} - \lambda_j \phi_j^{\otimes 3}$.
        end if
      end for
      Compute power iteration update $\theta_t^{(\tau)} := \frac{\widetilde{\mathcal{T}}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})}{\|\widetilde{\mathcal{T}}(I, \theta_{t-1}^{(\tau)}, \theta_{t-1}^{(\tau)})\|}$
    end for
  end for
  Let $\tau^* := \arg\max_{\tau\in L}\{\widetilde{\mathcal{T}}(\theta_N^{(\tau)}, \theta_N^{(\tau)}, \theta_N^{(\tau)})\}$.
  Do N power iteration updates starting from $\theta_N^{(\tau^*)}$ to obtain eigenvector estimate v_i, and set $\lambda_i := \widetilde{\mathcal{T}}(v_i, v_i, v_i)$.
end for
return the estimated eigenvalue/eigenvectors (λ, M).

v_i are orthonormal vectors and $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_k$. Let $\widehat{\mathcal{T}} = \mathcal{T} + E$ be the perturbed tensor with $\|E\| \leq \epsilon_T$. Recall that N denotes the number of iterations of the tensor power method. We call an initialization vector u to be (γ, R_0)-good if there exists v_i such that $\langle u, v_i \rangle > R_0$ and $|\langle u, v_i \rangle| - \max_{j<i} |\langle u, v_j \rangle| > \gamma |\langle u, v_i \rangle|$. Choose $\gamma = 1/100$.

Theorem 40 *There exists universal constants $C_1, C_2 > 0$ such that the following holds.*

$$\epsilon_T \leq C_1 \cdot \lambda_{\min} R_0^2, \qquad N \geq C_2 \cdot \left(\log(k) + \log\log\left(\frac{\lambda_{\max}}{\epsilon_T} \right) \right), \tag{A.2}$$

Assume there is at least one good initialization vector corresponding to each v_i, $i \in [k]$. The parameter ξ for choosing deflation vectors in each iteration of the tensor power method in Procedure 8 is chosen as $\xi \geq 25\epsilon_T$. We obtain eigenvalue-eigenvector pairs $(\hat{\lambda}_1, \hat{v}_1), (\hat{\lambda}_2, \hat{v}_2), \ldots, (\hat{\lambda}_k, \hat{v}_k)$ such that there exists a permutation η on $[k]$ with

$$\|v_{\eta(j)} - \hat{v}_j\| \leq 8\epsilon_T / \lambda_{\eta(j)}, \qquad |\lambda_{\eta(j)} - \hat{\lambda}_j| \leq 5\epsilon_T, \quad \forall j \in [k],$$

and

$$\left\| \mathcal{T} - \sum_{j=1}^{k} \hat{\lambda}_j \hat{v}_j^{\otimes 3} \right\| \leq 55\epsilon_T.$$

In the sequel, we establish concentration bounds that allows us to translate the above condition on tensor perturbation (A.2) to sample complexity bounds.

A.3.2 Concentration Bounds

Analysis of Whitening

Recall that we use the covariance operator $\mathcal{C}_{X_1 X_2}$ for whitening the 3rd order embedding $\mathcal{C}_{X_1, X_2, X_3}$. We first analyze the perturbation in whitening when sample estimates are employed.

Let $\widehat{\mathcal{C}}_{X_1X_2}$ denote the sample covariance operator between variables X_1 and X_2, and let

$$B := 0.5(\widehat{\mathcal{C}}_{X_1X_2} + \widehat{\mathcal{C}}_{X_1X_2}^\top) = \widehat{\mathcal{U}}\widehat{S}\widehat{\mathcal{U}}^\top$$

denote the SVD. Let $\widehat{\mathcal{U}}_k$ and $\widehat{S}_k$ denote the restriction to top-k eigen-pairs, and let $B_k := \widehat{\mathcal{U}}_k\widehat{S}_k\widehat{\mathcal{U}}_k^\top$. Recall that the whitening matrix is given by $\widehat{\mathcal{W}} := \widehat{\mathcal{U}}_k\widehat{S}_k^{-1/2}$. Now $\widehat{\mathcal{W}}$ whitens B_k, i.e. $\widehat{\mathcal{W}}^\top B_k\widehat{\mathcal{W}} = I$.

Now consider the SVD of

$$\widehat{\mathcal{W}}^\top\mathcal{C}_{X_1X_2}\widehat{\mathcal{W}} = ADA^\top,$$

and define

$$\mathcal{W} := \widehat{\mathcal{W}}AD^{-1/2}A^\top,$$

and $\mathcal{W}$ whitens $\mathcal{C}_{X_1X_2}$ since $\mathcal{W}^\top\mathcal{C}_{X_1X_2}W = I$. Recall that by exchangeability assumption,

$$\mathcal{C}_{X_1,X_2} = \sum_{j=1}^{k}\pi_j \cdot \mu_{X|j} \otimes \mu_{X|j} = M\,\mathrm{Diag}(\pi)M^\top, \tag{A.3}$$

where the j^{th} column of M, $M_j = \mu_{X|j}$.

We now establish the following perturbation bound on the whitening procedure. Recall from (A.13), $\epsilon_{pairs} := \left\|\mathcal{C}_{X_1,X_2} - \widehat{\mathcal{C}}_{X_1,X_2}\right\|$. Let $\sigma_1(\cdot) \geq \sigma_2(\cdot)\ldots$ denote the singular values of an operator.

Lemma 41 (Whitening perturbation) *Assuming that* $\epsilon_{pairs} < 0.5\sigma_k(\mathcal{C}_{X_1X_2})$,

$$\epsilon_W := \|\,\mathrm{Diag}(\pi)^{1/2}M^\top(\widehat{\mathcal{W}} - \mathcal{W})\| \leq \frac{4\epsilon_{pairs}}{\sigma_k(\mathcal{C}_{X_1X_2})} \tag{A.4}$$

Remark: Note that $\sigma_k(\mathcal{C}_{X_1X_2}) = \sigma_k^2(M)$.

Proof: The proof is along the lines of Lemma 16 of [25], but adapted to whitening using

the covariance operator here.

$$\begin{aligned}\|\operatorname{Diag}(\pi)^{1/2}M^\top(\widehat{\mathcal{W}}-\mathcal{W})\| &= \|\operatorname{Diag}(\pi)^{1/2}M^\top W(AD^{1/2}A^\top - I)\| \\ &\leq \|\operatorname{Diag}(\pi)^{1/2}M^\top\mathcal{W}\|\|D^{1/2}-I\|.\end{aligned}$$

Since $\mathcal{W}$ whitens $\mathcal{C}_{X_1X_2} = M\operatorname{Diag}(\pi)M^\top$, we have that $\|\operatorname{Diag}(\pi)^{1/2}M^\top\mathcal{W}\| = 1$. Now we control $\|D^{1/2}-I\|$. Let $\widetilde{E} := \mathcal{C}_{X_1,X_2} - B_k$, where recall that $B = 0.5(\widehat{\mathcal{C}}_{X_1,X_2} + \widehat{\mathcal{C}}^\top_{X_1X_2})$ and B_k is its restriction to top-k singular values. Thus, we have $\|\widetilde{E}\| \leq \epsilon_{pairs} + \sigma_{k+1}(B) \leq 2\epsilon_{pairs}$. We now have

$$\begin{aligned}\|D^{1/2}-I\| &\leq \|(D^{1/2}-I)(D^{1/2}+I)\| \leq \|D-I\| \\ &= \|ADA^\top - I\| = \|\widehat{\mathcal{W}}^\top\mathcal{C}_{X_1X_2}\widehat{\mathcal{W}} - I\| \\ &= \|\widehat{\mathcal{W}}^\top\widetilde{E}\widehat{\mathcal{W}}\| \leq \|\widehat{\mathcal{W}}\|^2(2\epsilon_{pairs}).\end{aligned}$$

Now

$$\|\widehat{\mathcal{W}}^2\| \leq \frac{1}{\sigma_k(\widehat{\mathcal{C}}_{X_1X_2})} \leq \frac{2}{\sigma_k(\mathcal{C}_{X_1X_2})},$$

when $\epsilon_{pairs} < 0.5\sigma_k(\mathcal{C}_{X_1X_2})$. □

Tensor Concentration Bounds

Recall that the whitened tensor from samples is given by

$$\widehat{\mathcal{T}} := \widehat{\mathcal{C}}_{X_1X_2X_3} \times_1 (\widehat{\mathcal{W}}^\top) \times_2 (\widehat{\mathcal{W}}^\top) \times_3 (\widehat{\mathcal{W}}^\top).$$

We want to establish its perturbation from the whitened tensor using exact statistics

$$\mathcal{T} := \mathcal{C}_{X_1X_2X_3} \times_1 (\mathcal{W}^\top) \times_2 (\mathcal{W}^\top) \times_3 (\mathcal{W}^\top).$$

Further, we have

$$\mathcal{C}_{X_1X_2X_3} = \sum_{h\in[k]} \pi_h \cdot \mu_{X|h} \otimes \mu_{X|h} \otimes \mu_{X|h} \tag{A.5}$$

Let $\epsilon_{triples} := \|\widehat{\mathcal{C}}_{X_1X_2X_3} - \mathcal{C}_{X_1X_2X_3}\|$. Let $\pi_{\min} := \min_{h\in[k]} \pi_h$.

Lemma 42 (Tensor perturbation bound) *Assuming that $\epsilon_{pairs} < 0.5\sigma_k(\mathcal{C}_{X_1X_2})$, we have*

$$\epsilon_T := \|\widehat{\mathcal{T}} - \mathcal{T}\| \leq \frac{2\sqrt{2}\epsilon_{triples}}{\sigma_k(\mathcal{C}_{X_1X_2})^{1.5}} + \frac{\epsilon_W^3}{\sqrt{\pi_{\min}}}. \tag{A.6}$$

Proof: Define intermediate tensor

$$\widetilde{\mathcal{T}} := \mathcal{C}_{X_1X_2X_3} \times_1 (\widehat{\mathcal{W}}^\top) \times_2 (\widehat{\mathcal{W}}^\top) \times_3 (\widehat{\mathcal{W}}^\top).$$

We will bound $\|\widehat{\mathcal{T}} - \widetilde{\mathcal{T}}\|$ and $\|\widehat{\mathcal{T}} - \mathcal{T}\|$ separately.

$$\|\widehat{\mathcal{T}} - \widetilde{\mathcal{T}}\| \leq \|\widehat{\mathcal{C}}_{X_1,X_2,X_2} - \mathcal{C}_{X_1,X_2,X_3}\| \|\widehat{\mathcal{W}}\|^3 \leq \frac{2\sqrt{2}\epsilon_{triples}}{\sigma_k(\mathcal{C}_{X_1X_2})^{1.5}},$$

using the bound on $\|\widehat{\mathcal{W}}\|$ in Lemma 41. For the other term, first note that

$$\mathcal{C}_{X_1,X_2,X_3} = \sum_{h\in[k]} \pi_h \cdot M_h \otimes M_h \otimes M_h,$$

$$\begin{aligned}
\|\widehat{\mathcal{T}} - \mathcal{T}\| &= \|\mathcal{C}_{X_1X_2X_3} \times_1 (\widehat{\mathcal{W}} - \mathcal{W})^\top \times_2 (\widehat{\mathcal{W}} - \mathcal{W})^\top \times_3 (\widehat{\mathcal{W}} - \mathcal{W})^\top\| \\
&\leq \frac{\|\operatorname{Diag}(\pi)^{1/2} M^\top (\widehat{\mathcal{W}} - \mathcal{W})\|^3}{\sqrt{\pi_{\min}}} \\
&= \frac{\epsilon_W^3}{\sqrt{\pi_{\min}}}
\end{aligned}$$

□

Proof of Theorem 2: We obtain a condition on the above perturbation ϵ_T in (A.6) by

applying Theorem 40 as $\epsilon_T \leq C_1 \lambda_{\min} R_0^2$. Here, we have $\lambda_i = 1/\sqrt{\pi_i} \geq 1$. For random initialization, we have that $R_0 \sim 1/\sqrt{k}$, with probability $1-\delta$ using $\mathrm{poly}(k)\,\mathrm{poly}(1/\delta)$ trials, see Thm. 5.1 in [10]. Thus, we require that $\epsilon_T \leq \frac{C_1}{k}$. Summarizing, we require for the following conditions to hold

$$\epsilon_{pairs} \leq 0.5\sigma_k(\mathcal{C}_{X_1X_2}), \quad \epsilon_T \leq \frac{C_1}{k}. \tag{A.7}$$

We now substitute for ϵ_{pairs} and $\epsilon_{triples}$ in (A.6) using Lemma 43 and Lemma 44.

From Lemma 43, we have that

$$\epsilon_{pairs} \leqslant \frac{2\sqrt{2}\rho\sqrt{\log\frac{2}{\delta}}}{\sqrt{m}},$$

with probability $1-\delta$. It is required that $\epsilon_{pairs} < 0.5\sigma_k(\mathcal{C}_{X_1,X_2})$, which yields that

$$m > \frac{32\rho^2 \log\frac{2}{\delta}}{\sigma_k^2(\mathcal{C}_{X_1,X_2})}. \tag{A.8}$$

Further we require that $\epsilon_T \leq C_1/k$, which implies that each of the terms in (A.6) is less than C/k, for some constant C. Thus, we have

$$\frac{2\sqrt{2}\epsilon_{triples}}{\sigma_k^{1.5}(\mathcal{C}_{X_1,X_2})} < \frac{C}{k} \quad \Rightarrow \quad m > \frac{C_3 k^2 \rho^3 \log\frac{2}{\delta}}{\sigma_k^3(\mathcal{C}_{X_1,X_2})},$$

for some constant C_3 with probability $1-\delta$ from Lemma 44. Similarly for the second term in (A.6), we have

$$\frac{\epsilon_W^3}{\sqrt{\pi_{\min}}} < \frac{C}{k},$$

and from Lemma 41, this implies that

$$\epsilon_{pairs} \leq \frac{C'\pi_{\min}^{1/6}\sigma_k(\mathcal{C}_{X_1,X_2})}{k^{1/3}},$$

Thus, we have

$$m > \frac{C_4 k^{\frac{2}{3}} \rho^2 \log \frac{2}{\delta}}{\pi_{\min}^{\frac{1}{3}} \sigma_k^2(\mathcal{C}_{X_1,X_2})},$$

for some other constant C_4 with probability $1-\delta$. Thus, we have the result in Theorem 2.

□

Concentration bounds for Empirical Operators

Concentration results for the singular value decomposition of empirical operators.

Lemma 43 (Concentration bounds for pairs) *Let $\rho := \sup_{x\in\Omega} k(x,x)$, and $\|\cdot\|$ be the Hilbert-Schmidt norm, we have for*

$$\epsilon_{pairs} := \left\| \mathcal{C}_{X_1X_2} - \widehat{\mathcal{C}}_{X_1X_2} \right\|, \tag{A.9}$$

$$\Pr\left\{ \epsilon_{pairs} \leqslant \frac{2\sqrt{2}\rho\sqrt{\log\frac{2}{\delta}}}{\sqrt{m}} \right\} \geqslant 1-\delta. \tag{A.10}$$

Proof We will use similar arguments as in [141] which deals with symmetric operator. Let ξ_i be defined as

$$\xi_i \;=\; \phi(x_1^i) \otimes \phi(x_2^i) - \mathcal{C}_{X_1,X_2}. \tag{A.11}$$

It is easy to see that $\mathbb{E}[\xi_i] = 0$. Further, we have

$$\sup_{x_1,x_2} \|\phi(x_1)\otimes\phi(x_2)\|^2 = \sup_{x_1,x_2} k(x_1,x_1)k(x_2,x_2) \leqslant \rho^2, \tag{A.12}$$

which implies that $\|\mathcal{C}_{X_1X_2}\| \leqslant \rho$, and $\|\xi_i\| \leqslant 2\rho$. The result then follows from the Hoeffding's inequality in Hilbert space. ■

Similarly, we have the concentration bound for 3rd order embedding.

Lemma 44 (Concentration bounds for triples) *Let $\rho := \sup_{x\in\Omega} k(x,x)$, and $\|\cdot\|$ be the Hilbert-Schmidt norm, we have for*

$$\epsilon_{triples} := \left\| \mathcal{C}_{X_1X_2X_3} - \widehat{\mathcal{C}}_{X_1X_2X_3} \right\|, \tag{A.13}$$

$$\Pr\left\{ \epsilon_{triples} \leqslant \frac{2\sqrt{2}\rho^{3/2}\sqrt{\log\frac{2}{\delta}}}{\sqrt{m}} \right\} \geqslant 1-\delta. \tag{A.14}$$

Proof We will use similar arguments as in [141] which deals with symmetric operator. Let ξ_i be defined as

$$\xi_i \;=\; \phi(x_1^i)\otimes\phi(x_2^i)\otimes\phi(x_3^i) - \mathcal{C}_{X_1X_2X_3}. \tag{A.15}$$

It is easy to see that $\mathbb{E}[\xi_i] = 0$. Further, we have

$$\sup_{x_1,x_2,x_3} \|\phi(x_1)\otimes\phi(x_2)\otimes\phi(x_3)\|^2 = \sup_{x_1,x_2,x_3} k(x_1,x_1)k(x_2,x_2)k(x_3,x_3) \leqslant \rho^3, \tag{A.16}$$

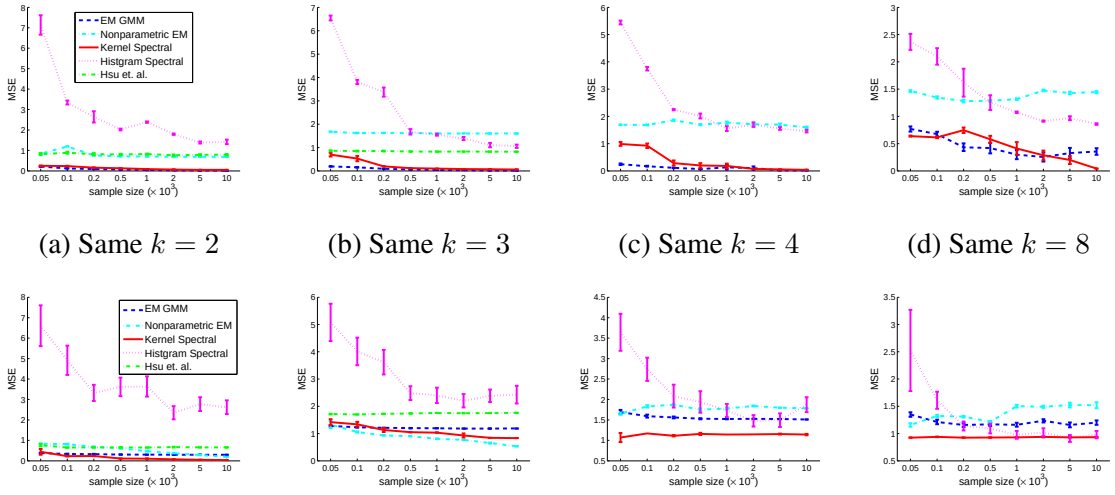
which implies that $\|\mathcal{C}_{X_1X_2X_3}\| \leqslant \rho^{3/2}$, and $\|\xi_i\| \leqslant 2\rho^{3/2}$. The result then follows from the Hoeffding's inequality in Hilbert space. ■

A.4 Experiment on Single Conditional Distribution

We also did some experiments for three-dimensional synthetic data that each view has the same conditional distribution. We generated the data from two settings:

1. Mixture of Gaussian conditional density;

2. Mixture of Gaussian and shifted Gamma conditional density.



(a) Same $k = 2$ (b) Same $k = 3$ (c) Same $k = 4$ (d) Same $k = 8$

(e) Diff. $k = 2$ (f) Diff. $k = 3$ (g) Diff. $k = 4$ (h) Diff. $k = 8$

Figure A.1: (a)-(d) Mixture of Gaussian distributions with $k = 2, 3, 4, 8$ components. (e)-(h) Mixture of Gaussian/Gamma distribution with $k = 2, 3, 4, 8$. For the former case, the performance of kernel spectral algorithm converge to those of EM algorithm for mixture of Gaussian model. For both cases, the performance of kernel spectral algorithm are consistently the best or comparable. Spherical Gaussian spectral algorithm does not work for $k = 4, 8$, and hence not plotted.

The mixture proportion and other experiment settings are exact same as the experiment in the main text. The only difference is that the conditional densities for each view here are the identical. We use the same measure to evaluate the performance. The empirical results are plotted in Figure A.1.

As we expected, the behavior of the proposed method is similar to the results in different conditional densities case. In mixture of Gaussians, our algorithm converges to the EM GMM resuls. And in the mixture of Gaussian/shift Gamma, our algorithm consistently better to other alternatives in most cases, except $k = 3$ where our method achieve comparable to nonparametric EM algorithm.

APPENDIX B

THEOREM PROOFS IN CHAPTER 4

B.1 Analysis Roadmap

In order to analyze the convergence of our doubly stochastic kernel PCA algorithm, we will need to define a few intermediate subspaces. For simplicity of notation, we will assume the mini-batch size for the data points is one.

1. Let $F_t := \left(f_t^1, \ldots, f_t^k\right)$ be the subspace estimated using stochastic gradient and explicit orthogonalization:

$$\begin{aligned} \tilde{F}_{t+1} &\leftarrow F_t + \eta_t A_t F_t \\ F_{t+1} &\leftarrow \tilde{F}_{t+1} \left(\tilde{F}_{t+1}^\top \tilde{F}_{t+1}\right)^{-1/2} \end{aligned} \tag{B.1}$$

2. Let $G_t := \left(g_t^1, \ldots, g_t^k\right)$ be the subspace estimated using stochastic update rule without orthogonalization:

$$G_{t+1} \leftarrow G_t + \eta_t \left(I - G_t G_t^\top\right) A_t G_t. \tag{B.2}$$

   where $A_t G_t$ and $G_t G_t^\top A_t G_t$ can be equivalently written using the evaluation of the function $\{g_t^i\}$ on the current data point, leading to the equivalent rule:

$$G_{t+1} \leftarrow G_t \left(I - \eta_t g_t g_t^\top\right) + \eta_t k(x_t, \cdot) g_t^\top. \tag{B.3}$$

3. Let $\tilde{G}_t := \left(\tilde{g}_t^1, \ldots, \tilde{g}_t^k\right)$ be the subspace estimated using stochastic update rule without orthogonalization, but the evaluation of the function $\{\tilde{g}_t^i\}$ on the current data

point is replaced by the evaluation $h_t = [h_t^i(x_t)]^\top$:

$$\tilde{G}_{t+1} \leftarrow \tilde{G}_t + \eta_t k(x_t, \cdot) h_t^\top - \eta_t \tilde{G}_t h_t h_t^\top \tag{B.4}$$

4. Let $H_t := \left(h_t^1, \ldots, h_t^k\right)$ be the subspace estimated using doubly stochastic update rule without orthogonalization, *i.e.*, the update rule:

$$H_{t+1} \leftarrow H_t + \eta_t \phi_{\omega_t}(x_t) \phi_{\omega_t}(\cdot) h_t^\top - \eta_t H_t h_t h_t^\top. \tag{B.5}$$

The relation of these subspaces are summarized in Table B.1. Using these notations, we describe a sketch of our analysis in the rest of the section, while the complete proofs are provided in the following sections.

We first consider the subspace G_t estimated using the stochastic update rule, since it is simpler and its proof can provide the bases for analyzing the subspace H_t estimated by the doubly stochastic update rule.

Table B.1: Relation between various subspaces.

Subspace	Evaluation	Orth.	Data Mini-batch	RF Mini-batch
V	–	–	–	–
F_t	$f_t(x)$	✓	✓	✗
G_t	$g_t(x)$	✗	✓	✗
$\tilde{G}_t$	$h_t(x)$	✗	✓	✗
H_t	$h_t(x)$	✗	✓	✓

B.1.1 Stochastic update

Our guarantee is on the cosine of the principal angle between the computed subspace and the ground truth eigen subspace V (also called the potential function), which is a standard criterion for measuring the quality of the subspace:

$$\cos^2 \theta(V, G_t) = \min_w \frac{\left\|V^\top G_t w\right\|^2}{\left\|G_t w\right\|^2}.$$

We will focus on the case when a good initialization V_0 is given:

$$V_0^\top V_0 = I, \;\; \cos^2 \theta(V, V_0) \geq 1/2. \tag{B.6}$$

In other words, we analyze the later stage of the convergence, which is typical in the literature (*e.g.*, [67]). The early stage can be analyzed using established techniques (*e.g.*, [65]).

We will also focus on the dependence of the potential function on the step t. For this reason, throughout the text we suppose $|k(x, x')| \leq \kappa, |\phi_\omega(x)| \leq \phi$ and regard κ and ϕ as constants. Note that this is true for all the kernels and corresponding random features considered. We further regard the eigengap $\lambda_k - \lambda_{k+1}$ as a constant, which is also true for typical applications and datasets. Details can be found in the following sections.

Our final guarantee for G_t is stated in the following.

Theorem 4 1 *Assume (B.6) and suppose the mini-batch sizes satisfy that for any* $1 \leq i \leq t$, $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$. *There exist step sizes* $\eta_i = O(1/i)$ *such that*

$$1 - \cos^2 \theta(V, G_t) = O(1/t).$$

The convergence rate $O(1/t)$ is in the same order as that when computing only the top eigenvector in linear PCA [65], though we are not aware of any other convergence rate for computing the top k eigenfunctions in Kernel PCA. The bound requires the mini-batch sizes are large enough so that the spectral norm of A is approximated up to the order of the eigengap. This is due to the fact that approximating A with A_t will result in an error term in the order of $\|A - A_t\|$, while the increase of the potential is in the order of the eigengap. Similar terms appear in the analysis of the noisy power method [68] which, however, requires normalization and is not suitable for the kernel case. We do not specify the mini-batch sizes, but by assuming suitable data distributions, it is possible to obtain explicit bounds; see for example [76, 77].

Proof sketch To prove the theorem, we first prove the guarantee for the normalized subspace F_t which is more convenient to analyze, and then show that the update rules for F_t and G_t are first order equivalent so that G_t enjoys the same guarantee.

Lemma 5 2 $1 - \cos^2\theta(V, F_t) = O(1/t)$.

Let c_t^2 denote $\cos^2\theta(V, F_t)$, then a key step in proving the lemma is to show that

$$c_{t+1}^2 \geq c_t^2(1 + 2\eta_t(\lambda_k - \lambda_{k+1} - 2\|A - A_t\|)(1 - c_t^2)) - O(\eta_t^2). \tag{B.7}$$

Therefore, we will need the mini-batch sizes large enough so that $2\|A - A_t\|$ is smaller than the eigen-gap.

Another key element in the proof of the theorem is the first order equivalence of the two update rules. To show this, we need to compare the subspaces obtained by applying the them on the same subspace G_t. So we introduce $F(G_t)$ to denote the subspace by applying the update rule of F_t on G_t:

$$\begin{aligned}
\tilde{F}(G_t) &\leftarrow G_t + \eta_t A_t G_t \\
F(G_t) &\leftarrow \tilde{F}(G_t)\left[\tilde{F}(G_t)^\top \tilde{F}(G_t)\right]^{-1/2}
\end{aligned}$$

We show that the potentials of G_{t+1} and $F(G_t)$ are close:

Lemma 6 3 $\cos^2\theta(V, G_{t+1}) = \cos^2\theta(V, F(G_t)) \pm O(\eta_t^2)$.

The lemma means that applying the two update rules to the same input will result in two subspaces with similar potentials. Since $\cos^2\theta(V, F(G_t))$ enjoys the recurrence in (B.7), we know that $\cos^2\theta(V, G_{t+1})$ also enjoys such a recurrence, which then results in $1 - \cos^2\theta(V, G_t) = O(1/t)$.

The proof of the lemma is based on the observation that

$$\cos^2\theta(V, X) = \lambda_{\min}(V^\top X(X^\top X)^{-1}X^\top V).$$

The lemma follows by plugging in $X = G_{t+1}$ or $X = F(G_t)$ and comparing their Taylor expansions w.r.t. η_t.

B.1.2 Doubly stochastic update

For doubly stochastic update rule, the computed H_t is no longer in the RKHS so the principal angle is not well defined. Since the eigenfunction v is usually used for evaluating on points x, we will use the following point-wise convergence in our analysis. For any function v in the subspace of V with unit norm $\|v\|_{\mathcal{F}} = 1$, we will find a specially chosen function h in the subspace of H_t such that for any x,

$$\text{err} := |v(x) - h(x)|^2$$

is small with high probability. More specifically, the w is chosen to be $\tilde{G}^\top v$, and let $\tilde{g} = \tilde{G}_t w$ and $h = H_t w$. Then the error measure can be decomposed as

$$\begin{aligned} |v(x) - h(x)|^2 &= |v(x) - \tilde{g}(x) + \tilde{g}(x) - h(x)|^2 \\ &\le 2\,|v(x) - \tilde{g}(x)|^2 + 2\,|\tilde{g}(x) - h(x)|^2 \\ &\le \underbrace{2\kappa^2\,\|v - \tilde{g}\|_{\mathcal{F}}^2}_{\text{(I: Lemma 8)}} + \underbrace{2\,|\tilde{g}(x) - h(x)|^2}_{\text{(II: Lemma 9)}}. \end{aligned} \tag{B.8}$$

The distance $\|v - \tilde{g}\|_{\mathcal{F}}$ is closely related to the squared sine of the subspace angle between V and $\tilde{G}_t$. In fact, by definition, $\|v - \tilde{g}\|_{\mathcal{F}}^2 = \|v\|_{\mathcal{F}}^2 - \|\tilde{g}\|_{\mathcal{F}}^2 \le 1 - \cos^2\theta(V, \tilde{G}_t)$. Therefore, the first error term can be bounded by the guarantee on $\tilde{G}_t$, which can be obtained by similar arguments as for the stochastic update case. For the second term, note that $\tilde{G}_t$ is defined in such a way that the difference between $\tilde{g}(x) = \tilde{G}_t(x)w$ and $h(x) = H_t(x)w$ is a martingale, which can be bounded by careful analysis.

Overall, we have the following results. Suppose we use random Fourier features; see [61]. Similar bounds hold for other random features, where the batch sizes will de-

pend on the concentration bound of the random features used.

Theorem 7 4 *Assume (B.6) and suppose the mini-batch sizes satisfy that for any* $1 \leq i \leq t$, $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$ *and are of order* $\Omega(\ln \frac{t}{\delta})$. *There exist step sizes* $\eta_i = O(1/i)$, *such that the following holds. If* $\Omega(1) = \lambda_k(\tilde{G}_i^\top \tilde{G}_i) \leq \lambda_1(\tilde{G}_i^\top \tilde{G}_i) = O(1)$ *for all* $1 \leq i \leq t$, *then for any* x *and any function* v *in the span of* V *with unit norm* $\|v\|_{\mathcal{F}} = 1$, *we have that with probability* $\geq 1 - \delta$, *there exists* h *in the span of* H_t *satisfying*

$$|v(x) - h(x)|^2 = O\left(\frac{1}{t} \ln \frac{t}{\delta}\right).$$

The point-wise error scales as $\tilde{O}(1/t)$ with the step t, which is in similar order as that for the stochastic update rule. Again, we require the spectral norm of A to be estimated up to the order of the eigengap, for the same reason as before. We additionally need that the random features approximate the kernel function up to constant accuracy on all the data points up to time t, since the evaluation of the kernel function on these points are used in the update. This eventually leads to $\Omega(\ln \frac{t}{\delta})$ mini-batch sizes. Finally, we need $\tilde{G}_i^\top \tilde{G}_i$ to be roughly isotropic, *i.e.*, $\tilde{G}_i$ is roughly orthonormal. Intuitively, this should be true for the following reasons: $\tilde{G}_0$ is orthonormal; the update for $\tilde{G}_t$ is close to that for G_t, which in turn is close to F_t that are orthonormal.

Proof sketch The analysis is carried out by bounding each term in (B.8) separately. As discussed above, in order to bound term I, we need a bound on the squared cosine of the subspace angle between V and $\tilde{G}_t$.

Lemma 8 5 $1 - \cos^2 \theta(V, \tilde{G}_t) = O\left(\frac{1}{t} \ln \frac{t}{\delta}\right)$.

To prove this lemma, we follow the argument for Theorem 4 and get the recurrence as shown in (B.7), except with an additional error term, which is caused by the fact that the update rule for $\tilde{G}_{t+1}$ is using the evaluation $h_t(x_t)$ rather than $\tilde{g}_t(x_t)$. Bounding this additional term thus relies on bounding the difference between $h_t(x) - \tilde{g}_t(x)$, which is

also what we need for bounding term II in (B.8). For this purpose, we show the following bound:

Lemma 9 6 *For any x and unit vector w, with probability $\geq 1-\delta$ over $(\mathcal{D}^t, \omega^t)$, $|\tilde{g}_t(x)w - h_t(x)w|^2 = O\left(\frac{1}{t}\ln\left(\frac{t}{\delta}\right)\right)$.*

The key to prove this lemma is that our construction of $\tilde{G}_t$ makes sure that the difference between $\tilde{g}_t(x)w$ and $h_t(x)w$ consists of their difference in each time step. Furthermore, the difference in each time step conditioned on previous history has mean 0. In other words, the difference forms a martingale and thus can be bounded by Azuma's inequality. The resulting bound depends on the mini-batch sizes, the step sizes η_i, and the evaluations $h_i(x_i)$ used in the update rules. We then judiciously choose the parameters and simplify it to the bound in the lemma. The complication of the proof is mostly due to the interweaving of the parameter values; see the following sections for the details.

B.2 Stochastic Update

To prove the convergence of the stochastic update rule, we first prove the convergence of the normalized version F_t, and then we establish the first-order equivalence of the potential functions of the two update rules for F_t and G_t. Since the final recurrence result does not depend on higher order terms, this first-order equivalence establishes the convergence of the stochastic update rule without normalization.

B.2.1 Stochastic update with normalization

We consider the potential function $1 - \cos^2\theta\,(V, F_t)$ and prove a recurrence for it. We first show this for the simpler case where at each step we use the expected operator A in the update rule (Lemma 45), and then show this for the general case where A_t can be different from A (Lemma 46). Then the bound in Lemma 5 follows from solving the recurrence in Lemma 46.

Update rule with expected operator

The following lemma states the recurrence for the update rule which replace A_t in the stochastic update rule with the expected operator $A = \mathbb{E}A_t$:

$$\begin{aligned} \tilde{F}_{t+1} &\leftarrow F_t + \eta_t A F_t \\ F_{t+1} &\leftarrow \tilde{F}_{t+1} \left(\tilde{F}_{t+1}^\top \tilde{F}_{t+1} \right)^{-1/2} \end{aligned} \tag{B.9}$$

Lemma 45 *Let the sequence $\{F_i\}_i$ be obtained from the update rule (B.9), then*

$$1 - \cos^2 \theta \left(V, F_{t+1} \right) \leq \left[1 - \cos^2 \theta \left(V, F_t \right) \right] \left[1 - 2\eta_t \left(\lambda_k - \lambda_{k+1} \right) \cos^2 \theta \left(V, F_t \right) \right] + \beta_t,$$

where $\beta_t = 5\eta_t^2 B^2 + 3\eta_t^3 B^3$ and λ_k and λ_{k+1} are the top k and $k+1$-th eigenvalues of A.

Proof First note that the cosine of subspace angle does not change under linear combination of the basis

$$\cos^2 \theta \left(V, F_{t+1} \right) = \min_{w'} \frac{\left\| V^\top F_{t+1} w' \right\|^2}{\left\| F_{t+1} w' \right\|^2} = \min_{w'} \frac{\left\| V^\top \tilde{F}_{t+1} \left(\tilde{F}_{t+1}^\top \tilde{F}_{t+1} \right)^{-1/2} w' \right\|^2}{\left\| \tilde{F}_{t+1} \left(\tilde{F}_{t+1}^\top \tilde{F}_{t+1} \right)^{-1/2} w' \right\|^2} = \min_{w} \frac{\left\| V^\top \tilde{F}_{t+1} w \right\|^2}{\left\| \tilde{F}_{t+1} w \right\|^2} \tag{B.10}$$

The update rule gives us

$$\left\| V^\top \tilde{F}_{t+1} w \right\|^2 \geq \left\| V^\top F_t w \right\|^2 + 2\eta_t \left\langle V^\top F_t w, V^\top A F_t w \right\rangle \tag{B.11}$$

$$\left\| \tilde{F}_{t+1} w \right\|^2 \leq \left\| F_t w \right\|^2 + 2\eta_t \left\langle F_t w, A F_t w \right\rangle + B \left\| F_t w \right\|^2 \eta_t^2 \tag{B.12}$$

Let $\hat{w} = w / \left\| F_t w \right\|$, $u = F_t \hat{w}$, so $\|u\| = 1$. Denote $c = \left\| V^\top u \right\|$ and $s = \left\| V_\perp^\top u \right\|$. According to the definition, we have $c \geq \cos \theta_k \left(V, F_t \right)$. Keep expanding the update rule

leads to

$$\begin{aligned}
\frac{\left\|V^\top \tilde{F}_{t+1} w\right\|^2}{\left\|\tilde{F}_{t+1} w\right\|^2} &\geq \frac{\left\|V^\top F_t w\right\|^2 + 2\eta_t \left\langle V^\top F_t w, V^\top A F_t w\right\rangle}{\left\|F_t w\right\|^2 + 2\eta_t \left\langle F_t w, A F_t w\right\rangle + B \left\|F_t w\right\|^2 \eta_t^2} \\
&= \frac{\left\|V^\top u\right\|^2 + 2\eta_t \left\langle V^\top u, V^\top A u\right\rangle}{1 + 2\eta_t \left\langle u, Au\right\rangle + B\eta_t^2} \\
&\geq \left\{\left\|V^\top u\right\|^2 + 2\eta_t \left\langle V^\top u, V^\top A u\right\rangle\right\} \left\{1 - 2\eta_t \left\langle u, Au\right\rangle - B\eta_t^2\right\} \\
&\geq \left\|V^\top u\right\|^2 + 2\eta_t \left\langle V^\top u, V^\top A u\right\rangle - 2\eta_t \left\|V^\top u\right\|^2 \left\langle u, Au\right\rangle \\
&- 5\eta_t^2 B^2 - 2\eta_t^3 B^3 \\
&= c^2 + 2\eta_t \left\{u^\top V V^\top A u - c^2 u^\top A u\right\} - \beta_t \\
&= c^2 + 2\eta_t u^\top \left(V V^\top - c^2 I\right) A u - \beta_t \\
&= c^2 + 2\eta_t u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) A u - \beta_t.
\end{aligned} \tag{B.13}$$

Recall that $A = V \Lambda_k V^\top + V_\perp \Lambda_{k+1} V_\perp^\top$. Then

$$\begin{aligned}
u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) A u &= s^2 u^\top V \Lambda_k V^\top u - c^2 u^\top V_\perp \Lambda_{k+1} V_\perp^\top u \\
&\geq \lambda_k s^2 c^2 - \lambda_{k+1} c^2 s^2 = s^2 c^2 \left(\lambda_k - \lambda_{k+1}\right)
\end{aligned} \tag{B.14}$$

The recurrence is therefore

$$\begin{aligned}
\cos^2 \theta \left(V, F_{t+1}\right) &\geq c^2 + 2\eta_t s^2 c^2 \left(\lambda_k - \lambda_{k+1}\right) - \beta_t \\
&= c^2 \left(1 + 2\eta_t \left(\lambda_k - \lambda_{k+1}\right) \left(1 - c^2\right)\right) - \beta_t.
\end{aligned} \tag{B.15}$$

The first term is a quadratic function of c^2:

$$x \left(1 + a \left(1 - x\right)\right) \tag{B.16}$$

where $x := c^2$ and $a = 2\eta_t \left(\lambda_k - \lambda_{k+1}\right)$. It has two roots at 0 and $1 + \frac{1}{a}$. Therefore, if

$\frac{1}{2} + \frac{1}{2a} \geq 1$, it is a monotonic increasing function in the interval of $[0, 1]$.

Thus, if $\eta_t \leq \frac{1}{4(\lambda_k - \lambda_{k+1})}$, which holds for all t large enough, we have

$$\cos^2 \theta\left(V, F_{t+1}\right) \geq \cos^2 \theta\left(V, F_t\right)\left(1 + 2\eta_t\left(\lambda_k - \lambda_{k+1}\right)\left(1 - \cos^2 \theta\left(V, F_t\right)\right)\right) - \beta_t \quad \text{(B.17)}$$

which leads to the lemma. ■

Using different operators in different iterations

Now consider the case of stochastic update rule (B.1) where we use a mini-batch to approximate the expectation in each iteration.

Lemma 46 *Let the sequence $\{F_i\}_i$ be obtained from the update rule (B.1), then*

$$1 - \cos^2 \theta\left(V, F_{t+1}\right) \leq \left[1 - \cos^2 \theta\left(V, F_t\right)\right]\left[1 - 2\eta_t\left(\lambda_k - \lambda_{k+1} - \|A_t - A\|\right)\cos^2 \theta\left(V, F_{t+1}\right)\right] + \beta_t,$$

where $\beta_t = 5\eta_t^2 B^2 + 3\eta_t^3 B^3$ and λ_k and λ_{k+1} are the top k and $k+1$-th eigenvalues of A.

Proof The effect of the stochastic update is an additional term in the recurrence

$$\cos^2 \theta\left(V, F_{t+1}\right) \geq c^2 + 2\eta_t u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) A u + Z_t - \beta_t \quad \text{(B.18)}$$

where

$$Z_t = 2\eta_t u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right)\left(A_t - A\right) u. \quad \text{(B.19)}$$

The effect of the noise can be bounded, i.e.

$$\begin{aligned} Z_t &= 2\eta_t s^2 u^\top V V^\top \left(A_t - A\right) u - 2\eta_t c^2 u^\top V_\perp V_\perp^\top \left(A_t - A\right) u \quad \text{(B.20)} \\ &= 2\eta_t s^2 u^\top \left(V V^\top + l_1 I\right)\left(A_t - A\right) u - 2\eta_t c^2 u^\top \left(V_\perp V_\perp^\top + l_2 I\right)\left(A_t - A\right) u, \end{aligned}$$

where $s^2 l_1 = c^2 l_2$ are positive numbers such that $VV^\top + l_1 I$ and $V_\perp V_\perp^\top + l_2 I$ are positive-definite.

The generalized Rayleigh quotient leads to the inequality

$$\begin{aligned} \left| u^\top \left(VV^\top + l_1 I \right) (A_t - A)\, u \right| &\leq \lambda u^\top \left(VV^\top + l_1 I \right) u \\ &\leq \lambda \left(c^2 + l_1 \right) \end{aligned} \tag{B.21}$$

where λ is the largest generalized eigen-value that satisfies

$$\left(VV^\top + l_1 I \right) (A_t - A)\, x = \lambda \left(VV^\top + l_1 I \right) x. \tag{B.22}$$

Since $VV^\top + l_1 I$ is positive definite, we have $\lambda = \|A_t - A\|$.

Similarly, we have

$$\left| u^\top \left(V_\perp V_\perp^\top + l_2 I \right) (A_t - A)\, u \right| \leq \|A_t - A\| \left(s^2 + l_2 \right). \tag{B.23}$$

The noise term is thus bounded by

$$Z_t \geq -2\eta_t s^2 \|A_t - A\| \left(c^2 + l_1 \right) - 2\eta_t c^2 \|A_t - A\| \left(s^2 + l_2 \right). \tag{B.24}$$

Note that l_1 and l_2 can be infinitely small positive so we can ignore them.

Therefore, the recurrence is

$$\begin{aligned} \cos^2 \theta \left(V, F_{t+1} \right) &\geq c^2 + 2\eta_t s^2 c^2 \left(\lambda_k - \lambda_{k+1} \right) - 4\eta_t \|A_t - A\| s^2 c^2 - \beta_t \\ &= c^2 \left(1 + 2\eta_t \left(\lambda_k - \lambda_{k+1} - 2 \|A_t - A\| \right) \left(1 - c^2 \right) \right) - \beta_t \end{aligned} \tag{B.25}$$

which then leads to the lemma. ■

In order to get fast convergence, we need to take sufficiently large mini-batches such that the variance of the noise is small enough compared with the eigen-gap.

B.2.2 Stochastic update without normalization

We show that the cosine angles of the two updates are first-order equivalent. Then, since the recurrence is not affected by higher order terms, when the step size is small enough, we can show it also converges in $O(1/t)$.

To show the first order equivalence, we need to compare the subspaces obtained by applying the them on the same subspace G_t. So we introduce $F(G_t)$ to denote the subspace by applying the update rule of F_t on G_t:

$$\begin{aligned} \tilde{F}(G_t) &\leftarrow G_t + \eta_t A_t G_t \\ F(G_t) &\leftarrow \tilde{F}(G_t) \left[\tilde{F}(G_t)^\top \tilde{F}(G_t) \right]^{-1/2} \end{aligned} \tag{B.26}$$

Then the first order equivalence as stated in Lemma 6 follows from the following two lemmas for the normalized update rule (B.1) and the unnormalized update rule (B.26), respectively.

Lemma 47 $\cos^2 \theta \left(V, F(G_t)\right) = \lambda_{min} \left(M + O(\eta^2)\right)$ *where*

$$M = V^\top P P^\top V + \eta V^\top P P^\top A V + \eta V^\top A P P^\top V - 2\eta V^\top P P^\top A P P^\top V,$$

where $P P^\top = G_t \left(G_t^\top G_t\right)^{-1} G_t^\top$*, and* P *is an orthonormal basis for the subspace* G_t*.*

Proof For simplicity, let G denote G_t, and let A denote A_t in the following. We first have

$$\cos^2\theta\left(V, F(G)\right) = \lambda_{\min}\left(V^\top F(G)F(G)^\top V\right) \tag{B.27}$$

$$= \lambda_{\min}\left(F(G)^\top VV^\top F(G)\right) \tag{B.28}$$

$$= \lambda_{\min}\left\{V^\top\left(G+\eta_t AG\right)\left[\left(G+\eta_t AG\right)^\top\left(G+\eta_t AG\right)\right]^{-1}\left(G+\eta_t AG\right)^\top V\right\}. \tag{B.29}$$

Note that (B.28) is due to the fact that

$$\begin{aligned}
\lambda_{\min}\left(F(G)^\top VV^\top F(G)\right) &= \min_w \frac{w^\top F(G)^\top VV^\top F(G)w}{w^\top w} \\
&= \min_w \frac{w^\top R^{-1}\left(G+\eta_t AG\right)^\top VV^\top\left(G+\eta_t AG\right)R^{-1}w}{w^\top w} \\
&= \min_z \frac{z^\top\left(G+\eta_t AG\right)^\top VV^\top\left(G+\eta_t AG\right)z}{z^\top R^2 z} \\
&= \min_z \frac{z^\top\left(G+\eta_t AG\right)^\top VV^\top\left(G+\eta_t AG\right)z}{z^\top\left(G+\eta_t AG\right)^\top\left(G+\eta_t AG\right)z} \\
&= \min_z \frac{\left\|V^\top\left(G+\eta_t AG\right)z\right\|^2}{\left\|\left(G+\eta_t AG\right)z\right\|^2}
\end{aligned}$$

where $R = \left[\left(G+\eta_t AG\right)^\top\left(G+\eta_t AG\right)\right]^{1/2}$.

Now turn back to (B.29). Expand the matrix-valued function

$$\phi(\eta) = \left[\left(G+\eta AG\right)^\top\left(G+\eta AG\right)\right]^{-1} \tag{B.30}$$

$$= \phi(0) + \phi'(0)\eta + O(\eta^2).$$

$$\phi'(0) = -2\left(G^\top G\right)^{-1} G^\top AG\left(G^\top G\right)^{-1}. \tag{B.31}$$

So,

$$\phi(\eta) = \left(G^\top G\right)^{-1} - 2\eta \left(G^\top G\right)^{-1} G^\top AG \left(G^\top G\right)^{-1} + O(\eta^2). \tag{B.32}$$

Therefore,

$$\begin{aligned}
&V^\top (G + \eta_t AG) \left[(G + \eta_t AG)^\top (G + \eta_t AG)\right]^{-1} (G + \eta_t AG)^\top V \qquad \text{(B.33)}\\
&= \left(V^\top G + \eta_t V^\top AG\right) \left[\left(G^\top G\right)^{-1} - 2\eta \left(G^\top G\right)^{-1} G^\top AG \left(G^\top G\right)^{-1} + O(\eta^2)\right] \left(G^\top V + \eta_t G^\top AV\right)\\
&= V^\top G \left(G^\top G\right)^{-1} G^\top V + \eta V^\top G \left(G^\top G\right)^{-1} G^\top AV + \eta V^\top AG \left(G^\top G\right)^{-1} G^\top V\\
&- 2\eta V^\top G \left(G^\top G\right)^{-1} G^\top AG \left(G^\top G\right)^{-1} G^\top V + O(\eta^2)\\
&= V^\top PP^\top V + \eta V^\top PP^\top AV + \eta V^\top APP^\top V - 2\eta V^\top PP^\top APP^\top V + O(\eta^2),
\end{aligned}$$

where $PP^\top = G\left(G^\top G\right)^{-1} G^\top$, and P is an orthonormal basis for the subspace G. ■

Lemma 48 $\cos^2\theta\left(V, G_{t+1}\right) = \lambda_{min}\left(M\right)$ *where* M *is as defined in Lemma 47.*

Proof For simplicity, let G denote G_t and let A denote A_t. Then $\cos^2\theta\left(V, G_{t+1}\right) = \lambda_{\min}\left(N\right)$, where

$$N = V^\top G_{t+1} \left[G_{t+1}^\top G_{t+1}\right]^{-1} G_{t+1}^\top V \text{ with } G_{t+1} = G + \eta\left(I - GG^\top\right) AG.$$

Now it suffices to show $N = M$. Consider

$$\phi(\eta) = \left[\left(G + \eta\left(I - GG^\top\right) AG\right)^\top \left(G + \eta\left(I - GG^\top\right) AG\right)\right]^{-1}.$$

Then

$$\phi'(0) = -\left(G^\top G\right)^{-1} \left[G^\top \left(I - GG^\top\right) AG + G^\top A\left(I - GG^\top\right) G\right] \left(G^\top G\right)^{-1}$$

Therefore, N is

$$
\begin{aligned}
& V^\top \left(G+\eta\left(I-GG^\top\right)AG\right)\left[\left(G+\eta\left(I-GG^\top\right)AG\right)^\top\left(G+\eta\left(I-GG^\top\right)AG\right)\right]^{-1} \\
& \times\left(G+\eta\left(I-GG^\top\right)AG\right)^\top V \\
= \; & \left(V^\top G+\eta V^\top\left(I-GG^\top\right)AG\right)\left[\left(G+\eta\left(I-GG^\top\right)AG\right)^\top\left(G+\eta\left(I-GG^\top\right)AG\right)\right]^{-1} \\
& \times\left(G^\top V+\eta G^\top A\left(I-GG^\top\right)V\right) \\
= \; & \left(V^\top G+\eta V^\top\left(I-GG^\top\right)AG\right) \\
& \times\left[\left(G^\top G\right)^{-1}-\eta\left(G^\top G\right)^{-1}\left[G^\top\left(I-GG^\top\right)AG+G^\top A\left(I-GG^\top\right)G\right]\left(G^\top G\right)^{-1}\right] \\
& \times\left(G^\top V+\eta G^\top A\left(I-GG^\top\right)V\right) \\
= \; & V^\top G\left(G^\top G\right)^{-1}G^\top V+\eta V^\top G\left(G^\top G\right)^{-1}G^\top A\left(I-GG^\top\right)V+\eta V^\top\left(I-GG^\top\right)AG\left(G^\top G\right)^{-1}G \\
& -\eta V^\top G\left(G^\top G\right)^{-1}\left[G^\top\left(I-GG^\top\right)AG+G^\top A\left(I-GG^\top\right)G\right]\left(G^\top G\right)^{-1}G^\top V \\
= \; & V^\top PP^\top V+\eta V^\top PP^\top A\left(I-GG^\top\right)V+\eta V^\top\left(I-GG^\top\right)APP^\top V \\
& -\eta V^\top PP^\top\left(I-GG^\top\right)APP^\top V-\eta V^\top PP^\top A\left(I-GG^\top\right)PP^\top V \\
= \; & V^\top PP^\top V+\eta V^\top PP^\top AV+\eta V^\top APP^\top V-2\eta V^\top PP^\top APP^\top V \\
& -\eta V^\top PP^\top AGG^\top V-\eta V^\top GG^\top APP^\top V+\eta V^\top PP^\top GG^\top APP^\top V+\eta V^\top PP^\top AGG^\top PP^\top V \\
= \; & V^\top PP^\top V+\eta V^\top PP^\top AV+\eta V^\top APP^\top V-2\eta V^\top PP^\top APP^\top V
\end{aligned}
$$

which completes the proof. ■

B.3 Doubly Stochastic Update

In this section, we consider the doubly stochastic update rule. Suppose in step t, we use a mini-batch consisting of $B_{x,t}$ random data points $x_t^r (1 \leq r \leq B_{x,t})$ and $B_{\omega,t}$ random

features $\omega_t^s (1 \le s \le B_{\omega,t})$. Then the update rule is

$$H_{t+1} = H_t + \eta_t \mathbb{E}_t \left[\phi_{\omega_t}(x_t)\phi_{\omega_t}(\cdot)h_t(x_t)\right] - \eta_t H_t \mathbb{E}_t \left[h_t(x_t)^\top h_t(x_t)\right] \tag{B.34}$$

$$= H_t(I - \eta_t \mathbb{E}_t \left[h_t(x_t)^\top h_t(x_t)\right]) + \eta_t \mathbb{E}_t \left[\phi_{\omega_t}(x_t)\phi_{\omega_t}(\cdot)h_t(x_t)\right] \tag{B.35}$$

where for any function $f(x, \omega)$, $\mathbb{E}_t f(x_t, \omega)$ denotes $\sum_{r=1}^{B_{x,t}} \sum_{s=1}^{B_{\omega,t}} f(x_t^r, \omega_t^s)/(B_{x,t}B_{\omega,t})$. As before, we assume $H_0 = F_0$ is a good initialization, *i.e.*, $F_0^\top F_0 = I$ and $\cos^2 \theta(F_0, V) \ge 1/2$. Note that $H_t = [h_t^1(\cdot), \ldots, h_t^k(\cdot)]$, while $h_t(x_t)$ is its evaluation at x_t, *i.e.*, $h_t(x_t)$ is a row vector $[h_t^1(x_t), \ldots, h_t^k(x_t)]$.

We introduce the following intermediate function for analysis:

$$\tilde{G}_{t+1} = \tilde{G}_t + \eta_t \mathbb{E}_t \left[k(x_t, \cdot)h_t(x_t)\right] - \eta_t \tilde{G}_t \mathbb{E}_t \left[h_t(x_t)^\top h_t(x_t)\right] \tag{B.36}$$

$$= \tilde{G}_t(I - \eta_t \mathbb{E}_t \left[h_t(x_t)^\top h_t(x_t)\right]) + \eta_t \mathbb{E}_t \left[k(x_t, \cdot)h_t(x_t)\right]. \tag{B.37}$$

Again, $\tilde{G}_0 = F_0$.

The analysis follows our intuition: we first bound the difference between H_t and $\tilde{G}_t$ by a martingale argument, and then bound the difference between $\tilde{G}_t$ and V. For the second step we can apply the previous argument. Note that $\tilde{G}_t$ is different from F_t since $A_t F_t = k(x_t, \cdot)F_t(x_t)$ is now replaced by $k(x_t, \cdot)h_t(x_t)$, so we need to adjust our previous analysis.

Suppose we use random Fourier features for points in $\mathbb{R}^d$; see [61]. Then we have

Theorem 7 7 *Suppose the mini-batch sizes satisfy that for any $1 \le i \le t$, $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$ and $B_{x,i} = \Omega(\ln \frac{t}{\delta})$. There exist step sizes $\eta_i = O(1/i)$, such that the following holds. If $\Omega(1) = \lambda_k(\tilde{G}_i^\top \tilde{G}_i) \le \lambda_1(\tilde{G}_i^\top \tilde{G}_i) = O(1)$ for all $1 \le i \le t$, then for any x and any function v in the span of V with unit norm $\|v\|_{\mathcal{F}} = 1$, we have that with probability $\ge 1 - \delta$, there exists h in the span of H_t satisfying*

$$|v(x) - h(x)|^2 = O\left(\frac{1}{t} \ln \frac{t}{\delta}\right).$$

Proof Let $w = \tilde{G}_t^\top v$, $\tilde{g} = \tilde{G}_t w$, and $h = H_t w$.

$$\begin{aligned}
|v(x) - h(x)|^2 &= |v(x) - \tilde{g}(x) + \tilde{g}(x) - h(x)|^2 \\
&\le 2\,|v(x) - \tilde{g}(x)|^2 + 2\,|\tilde{g}(x) - h(x)|^2 \\
&\le 2\,\|v - \tilde{g}\|_{\mathcal{F}}^2\,\|k(x,\cdot)\|_{\mathcal{F}}^2 + 2\,|\tilde{g}(x) - h(x)|^2 \\
&\le 2\kappa^2\,\|v - \tilde{g}\|_{\mathcal{F}}^2 + |\tilde{g}(x) - h(x)|^2 .
\end{aligned}$$

Roughly speaking, the difference between v and $\tilde{g}$ is the error due to random data points and can be bounded by Lemma 52, while the difference between $\tilde{g}(x)$ and $h(x)$ is the error due to random features and can be bounded by Lemma 50. More precisely, since $\tilde{g}$ is the projection of v on the span of $\tilde{G}_t$,

$$\|v - \tilde{g}\|_{\mathcal{F}}^2 = \|v\|_{\mathcal{F}}^2 - \|\tilde{g}\|_{\mathcal{F}}^2 \le 1 - \cos^2\theta(\tilde{G}_t, V) = O\left(\frac{1}{t}\ln\frac{t}{\delta}\right)$$

where the last step is by Lemma 52. Also, since $\|w\| \le 1$, we have $|\tilde{g}(x) - h(x)|^2 = O\left(\frac{1}{t}\ln\frac{t}{\delta}\right)$ by Lemma 50.

What is left is to check the mini-batch sizes; see the assumptions in Lemma 49 and Lemma 52. We need $\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right]) = \lambda_k(\mathbb{E}_x\left[h_i(x)^\top h_i(x)\right]) \pm O(1)$, so we only need to estimate $\mathbb{E}_x\left[h_i^j(x)^\top h_i^\ell(x)\right]$ up to constant accuracy for all $1 \le j, \ell \le k$, for which $B_{x,i} = O(\ln\frac{t}{\delta})$ suffices. We also need $\Delta_\omega = O(\lambda_k - \lambda_{k+1}) = O(1)$, so we only need $\Delta_\omega = O(1)$. This is a bound for $(tB_{x,i})^2$ pairs of points, for which $B_{\omega,i} = O(\ln\frac{t}{\delta})$ suffices. ■

Similar bounds hold for other random features, where the batch sizes will depend on the concentration bound of the random features used.

The rest of this section is the proof of the theorem. For simplicity, $\|\cdot\|_{\mathcal{F}}$ is shorten as $\|\cdot\|$.

First, we bound the difference between H_t and $\tilde{G}_t$.

Lemma 49 *Suppose* $|k(x,x')| \leq \kappa, |\phi_\omega(x)| \leq \phi$. *Suppose the mini-batch sizes are large enough so that* $\left|k(x_i,x_j) - \sum_{s=1}^{B_{\omega,i}} \phi_{\omega_s}(x_i)\phi_{\omega_s}(x_j)/B_{\omega,i}\right| \leq \Delta_\omega$ *for all sampled data points* x_i *and* x_j. *For any* w *and* x, *with probability* $\geq 1-\delta$ *over* $(\mathcal{D}^t, \omega^t)$,

$$|\tilde{g}_{t+1}(x)w - h_{t+1}(x)w|^2 \leq B_{t+1}^2 := \frac{1}{2}\Delta_\omega^2 \ln\left(\frac{2}{\delta}\right) \sum_{i=1}^{t} \left|\mathbb{E}_i \left|h_i(x_i)\right| a_{t,i} w\right|^2$$

where $a_{t,i} = \eta_i \prod_{j=i+1}^{t} \left(I - \eta_j \mathbb{E}_j \left[h_j(x_j)^\top h_j(x_j)\right]\right)$ *for* $1 \leq i \leq t$, *and* $|h_i(x_i)| := \left[\left|h_i^j(x_i)\right|\right]_{j=1}^k$.

Proof Note that

$$H_{t+1} = \sum_{i=1}^{t} \mathbb{E}_i \left[\phi_{\omega_i}(x_i)\phi_{\omega_i}(\cdot)h_i(x_i)\right] a_{t,i} + F_0 a_{t,0}, \tag{B.38}$$

$$\tilde{G}_{t+1} = \sum_{i=1}^{t} \mathbb{E}_i \left[k(x_i,\cdot)h_i(x_i)\right] a_{t,i} + F_0 a_{t,0}, \tag{B.39}$$

where $a_{t,0} = \prod_{j=1}^{t} \left(I - \eta_j \mathbb{E}_j \left[h_j(x_j)^\top h_j(x_j)\right]\right)$.

We have $\tilde{g}_{t+1}(x)w - h_{t+1}(x)w = \sum_{i=1}^{t} V_{t,i}(x)$ where

$$V_{t,i}(x) = \mathbb{E}_i \left[k(x_i,x)h_i(x_i) - \phi_{\omega_i}(x_i)\phi_{\omega_i}(x)h_i(x_i)\right] a_{t,i} w.$$

$V_{t,i}(x)$ is a function of $(\mathcal{D}^i, \omega^i)$ and

$$\mathbb{E}_{\mathcal{D}^i,\omega^i}\left[V_{t,i}(x)|\omega^{i-1}\right] = \mathbb{E}_{\mathcal{D}^i,\omega^{i-1}}\mathbb{E}_{\omega_i}\left[V_{t,i}(x)|\omega^{i-1}\right] = 0,$$

so $\{V_{t,i}(x)\}$ is a martingale difference sequence.

Since $|V_{t,i}(x)| < \Delta_\omega |\mathbb{E}_i \left|h_i(x_i)\right| a_{t,i} w|$, the lemma follows from Azuma's Inequality. ■

So to bound $|\tilde{g}_t(x)w - h_t(x)w|$, we need to bound $|\mathbb{E}_i \left|h_i(x_i)\right| a_{t,i} w|$, which requires some additional assumptions.

Lemma 50 (Complete version of Lemma 9) *Suppose the conditions in Lemma 49 are true. Further suppose for all $i \le t$, $\eta_i = \theta/i$ where θ is sufficiently large so that $\theta\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right]) \ge 1$; also suppose $\lambda_1\left(\tilde{G}_i^\top \tilde{G}_i\right) = O(1)$.*

(1) With probability $\ge 1-\delta$ over $(\mathcal{D}^t, \omega^t)$, for all $1 \le i \le t$ and $\ell \in [k]$, we have

$$|\tilde{g}_i^\ell(x_i) - h_i^\ell(x_i)|^2 = O\left(\frac{\Delta_\omega^2\theta^4}{t}\ln\left(\frac{t}{\delta}\right)\right).$$

(2) For any x and unit vector w, with probability $\ge 1-\delta$ over $(\mathcal{D}^t, \omega^t)$,

$$|\tilde{g}_t(x)w - h_t(x)w|^2 = O\left(\frac{\Delta_\omega^2\theta^4}{t}\ln\left(\frac{t}{\delta}\right)\right).$$

Proof We first do induction on statement (1), which is true initially. Assume it is true for t, we prove it for $t+1$.

We have that for any unit vector w,

$$\begin{aligned}
|\mathbb{E}_i\left|h_i(x_i)\right| a_{t,i} w| &= \left|\eta_i \mathbb{E}_i\left|h_i(x_i)\right| \prod_{j=i+1}^{t}\left[I - \eta_j \mathbb{E}_j\left[h_j(x_j)^\top h_j(x_j)\right]\right] w\right| \\
&\le \eta_i \left\|\mathbb{E}_i\left|h_i(x_i)\right|\right\| \|w\| \prod_{j=i+1}^{t}\left\|I - \eta_j \mathbb{E}_j\left[h_j(x_j)^\top h_j(x_j)\right]\right\| \\
&\le O(1)\frac{\theta^2}{i}\prod_{j=i+1}^{t}\left(1-\frac{1}{j}\right) = O\left(\frac{\theta^2}{t}\right).
\end{aligned}$$

We use in the second line

$$\|h_i(x_i)\| \le O\left(\sqrt{\frac{\theta^2}{t}\ln\frac{t}{\delta}}\right) + \|\tilde{g}_i(x_i)\| \le O\left(\sqrt{\frac{\theta^2}{t}\ln\frac{t}{\delta}}\right) + \sqrt{\left\|\tilde{G}_i^\top \tilde{G}_i\right\|}\,\|\phi(x_i)\| = O(\theta)$$

that holds with probability $1 - t\delta/(t+1)$ by induction, and we use in the last line $\theta\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right]) \ge 1$.

Then by Lemma 49, with probability $\geq 1 - \delta/(k(t+1))$,

$$|\tilde{g}_{t+1}(x_{t+1})w - h_{t+1}(x_{t+1})w|^2 \leq \frac{1}{2}\Delta_\omega^2 \ln\left(\frac{2(t+1)}{\delta}\right) \sum_{i=1}^{t} \left|\mathbb{E}_i \left|h_i(x_i)\right| a_{t,i} w\right|^2$$
$$\leq O(\Delta_\omega^2) \ln\left(\frac{t+1}{\delta}\right) \sum_{i=1}^{t} \frac{\theta^4}{t^2} = O\left(\frac{\Delta_\omega^2 \theta^4}{t+1} \ln\left(\frac{t+1}{\delta}\right)\right).$$

Repeating the argument for k basis vectors $w = e_i (1 \leq i \leq k)$ completes the proof.

The other statement follows from similar arguments. ■

Next, we bound the difference between $\tilde{G}_t$ and V.

Lemma 51 *Suppose the conditions in Lemma 50 are true and furthermore,* $\lambda_k(\tilde{G}_i^\top \tilde{G}_i) = \Omega(1)$ *for all* $i \in [t]$. *Let* c_t^2 *denote* $\cos^2\theta(\tilde{G}_t, V)$. *Then with probability* $\geq 1 - \delta$,

$$c_{t+1}^2 \geq c_t^2 \left\{1 + 2\eta_t \left[\lambda_k - \lambda_{k+1} - 2\left\|A_t - A\right\| - O\left(\Delta_\omega \theta^2 \sqrt{\frac{1}{t} \ln \frac{t}{\delta}}\right)\right] \left(1 - c_t^2\right) - O\left(\eta_t \Delta_\omega \theta^2 \sqrt{\frac{1 - c_t^2}{t}} \ln\right.\right.$$

where β_t *is as defined in Lemma 45.*

Proof The potential of $\tilde{G}_t$ can be computed by a similar argument as in the previous section; the only difference is replacing $A_t u$ with $k(x_t, \cdot) h_t(x_t) \widehat{w}$. This leads to

$$\cos^2\theta(\tilde{G}_{t+1}, V) \geq c^2 + 2\eta_t u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) k(x_t, \cdot) h_t(x_t) \widehat{w} - \beta_t$$
$$= c^2 + 2\eta_t u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) \left[\left(k(x_t, \cdot) h_t(x_t) \widehat{w} - A_t u\right) + \left(A_t u - A u\right) + A u\right] - \beta_t \tag{B.40}$$

where $u = \tilde{G}_t \widehat{w}$ with unit norm $\|u\| = 1$.

The terms involving $(A_t u - A u)$ and $A u$ can be dealt with as before, so we only need

to bound the extra term

$$
\begin{aligned}
& u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) \left[k(x_t, \cdot) h_t(x_t) \widehat{w} - A_t u\right] \\
= & u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) \left[k(x_t, \cdot) h_t(x_t) \widehat{w} - k(x_t, \cdot) \tilde{g}_t(x_t) \widehat{w}\right] \\
= & u^\top \left(s^2 V V^\top - c^2 V_\perp V_\perp^\top\right) k(x_t, \cdot) [h_t(x_t) - \tilde{g}_t(x_t)] \widehat{w}.
\end{aligned}
$$

So we need to bound $[h_t(x_t) - \tilde{g}_t(x_t)]\widehat{w}$, which in turn relies on Lemma 50. More precisely, we have $\|h_t(x_t) - \tilde{g}_t(x_t)\|_\infty \leq \tilde{O}\left(\Delta_\omega \theta^2 \sqrt{1/t}\right)$ with probability $\geq 1 - \delta$. Also, we have $u = \tilde{G}_t \widehat{w}$ has unit norm, so $\|\widehat{w}\| = O(1)$ when $\lambda_k(\tilde{G}_i^\top \tilde{G}_i) = \Omega(1)$. Then

$$
\left|u^\top V V^\top k(x_t, \cdot)[h_t(x_t) - \tilde{g}_t(x_t)]\widehat{w}\right| \leq \left\|u^\top V\right\| \|k(x_t, \cdot)\| \tilde{O}\left(\Delta_\omega \theta^2 \sqrt{1/t}\right) \leq c^2 \tilde{O}\left(\Delta_\omega \theta^2 \sqrt{1/t}\right)
$$

where the last step follows from $c \geq 1/2$ by assumption. Similarly,

$$
\left|u^\top V_\perp V_\perp^\top k(x_t, \cdot)[h_t(x_t) - \tilde{g}_t(x_t)]\widehat{w}\right| \leq \left\|u^\top V_\perp\right\| \|k(x_t, \cdot)\| \tilde{O}\left(\Delta_\omega \theta^2 \sqrt{1/t}\right) \leq s \tilde{O}\left(\Delta_\omega \theta^2 \sqrt{1/t}\right) = \tilde{O}\left(\Delta
$$

Plugging into (B.40) and apply a similar argument as in Lemma 45 and 46 we have the lemma. ■

Lemma 52 (Complete version of Lemma 8) *If the mini-batch sizes are large enough so that* $\|A - A_i\| < (\lambda_k - \lambda_{k+1})/8$, $\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right]) = \lambda_k(\mathbb{E}_x\left[h_i(x)^\top h_i(x)\right]) \pm O(1)$, *and* $\Delta_\omega = O(\lambda_k - \lambda_{k+1})$, *then*

(1) $\theta = O(1)$;

(2) $1 - c_t^2 = O\left(\frac{1}{t} \ln \frac{t}{\delta}\right)$.

Proof If the mini-batch size is large enough so that $\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right]) = \lambda_k(\mathbb{E}_x\left[h_i(x)^\top h_i(x)\right]) \pm O(1)$, we only need to show $\lambda_k(\mathbb{E}_x\left[h_i(x)^\top h_i(x)\right]) = \Omega(1)$, which will lead to $\theta = O(1)$ and then solving the recurrence in Lemma 51 leads to $1 - \cos^2\theta(\tilde{G}_{t+1}, V) = \tilde{O}(1/t)$.

Let $e_i(x) = h_i(x) - \tilde{g}_i(x)$. Then

$$\mathbb{E}_x\left[h_i(x)^\top h_i(x)\right] = \mathbb{E}_x\left[\tilde{g}_i(x)^\top \tilde{g}_i(x)\right] + 2\mathbb{E}_x\left[e_i(x)^\top h_i(x)\right] - \mathbb{E}_x\left[e_i(x)^\top e_i(x)\right].$$

By Lemma 50, $\mathbb{E}_x\left|e_i^j(x)\right| = \tilde{O}(\theta^4/t)$, which is $o(1)$ if $\theta = O(1)$. Then the norm of $2\mathbb{E}_x\left[e_i(x)^\top h_i(x)\right] - \mathbb{E}_x\left[e_i(x)^\top e_i(x)\right]$ is $o(1)$, so we only need to consider $\mathbb{E}_x\left[\tilde{g}_i(x)^\top \tilde{g}_i(x)\right]$.

Formally, we prove our statements (1)(2) by induction. They are true initially. Suppose they are true for $t-1$, we prove them for t.

First, by solving the recurrence for c_t, we have that statement (2) is true up to step t.

Next, since $\mathbb{E}_x\left[\tilde{g}_t(x)^\top \tilde{g}_t(x)\right] = \tilde{G}_t^\top A \tilde{G}_t$, we have

$$\begin{aligned} w^\top \mathbb{E}_x\left[\tilde{g}_t(x)^\top \tilde{g}_t(x)\right] w =& w^\top \tilde{G}_t^\top A \tilde{G}_t w \\ =& w^\top \tilde{G}_t^\top (V \Lambda_k V^\top + V_\perp \Lambda_\perp V_\perp^\top) \tilde{G}_t w \\ \geq& w^\top \tilde{G}_t^\top V \Lambda_k V^\top \tilde{G}_t w \\ \geq& \lambda_k c_t^2 \left\|w\right\|^2 \end{aligned}$$

which means $\lambda_k(\mathbb{E}_x\left[\tilde{g}_t(x)^\top \tilde{g}_t(x)\right]) = \Omega(1)$ by induction on c_t and by the assumption that $\lambda_k(\tilde{G}_t^\top \tilde{G}_t) = \Omega(1)$. This then leads to $\lambda_k(\mathbb{E}_i\left[h_i(x_i)^\top h_i(x_i)\right])) = \Omega(1)$, which means $\theta = O(1)$ up to step t. ■

APPENDIX C

THEOREM PROOFS IN CHAPTER 6

C.1 Spherical harmonic decomposition and kernel spectrum

Any function defined on the unit sphere has a spherical harmonic decomposition

$$g(x,y) = \sum_u \gamma_u \phi_u(x)\phi_u(y), \tag{C.1}$$

where $\phi_u(x) : \mathbb{S}^{d-1} \mapsto \mathbb{R}$ is a spherical harmonic basis. Note that $u = (t,j)$ is a multi-index: the first denotes the order of the basis and the latter denotes the index of bases with the same order.

For each order t, there are $N(d,t) = \frac{2t+d-2}{t}\begin{pmatrix} t+d-3 \\ t-1 \end{pmatrix}$ bases with the same coefficient. As a result, the spectrum γ_u sorted by magnitude has the step like shape where each step is of length $N(d,t)$.

To compute the coefficients, we use the Legendre harmonics [142] with the following property

$$P_{t,d}(\langle x,y\rangle) = \frac{1}{N(d,t)} \sum_{j=1}^{N(d,t)} \phi_{t,j}(x)\phi_{t,j}(y). \tag{C.2}$$

The spherical harmonics also form an orthonormal basis on the unit sphere:

$$\mathbb{E}\left[\phi_{l,i}(x)\phi_{m,j}(x)\right] = \frac{1}{|\mathbb{S}^{d-1}|}\int_{\mathbb{S}^{d-1}} \phi_{l,i}(x)\phi_{m,j}(x)dx = \delta_{lm}\delta_{ij}, \tag{C.3}$$

where $\left|\mathbb{S}^{d-1}\right|$ denotes the surface area of the unit sphere.

Combining these properties, we can calculate the spectrum using

$$\gamma_{(t,j)} = \frac{\left|\mathbb{S}^{d-2}\right|}{\left|\mathbb{S}^{d-1}\right|} \int_{-1}^{1} g(\xi) P_{t,d}(\xi)(1-\xi^2)^{(d-3)/2} d\xi, \quad \text{for all } j \in [N(d,t)]\,. \tag{C.4}$$

C.2 Bounding $\lambda_m(G)$ using matrix concentration bound: Proof of Lemma 28

Recall that

$$g(x,y) = \sum_{u=1}^{\infty} \gamma_u \phi_u(x)\phi_u(y). \tag{C.5}$$

For an integer $r > 0$, define the truncated version of g and the corresponding residue as

$$g^{[r]}(x,y) = \sum_{u=1}^{r} \gamma_u \phi_u(x)\phi_u(y) \tag{C.6}$$

$$e^r(x,y) = g(x,y) - g^{[r]}(x,y) \tag{C.7}$$

and define the matrices

$$\left[G^{[r]}\right]_{i,j} = g^{[r]}(x_i, x_j)$$
$$E^r = G - G^{[r]}. \tag{C.8}$$

Lemma 53 *Let $c_g = \max_x g(x,x)$ then with probability at least $1 - m\exp\left(-\frac{m\gamma_m}{8c_g}\right)$,*

$$\lambda_m(G^{[m]}) \geq m\gamma_m/2.$$

Proof Define a matrix A whose rows are

$$A^i := [\sqrt{\gamma_1}\phi_1(x_i), \ldots, \sqrt{\gamma_m}\phi_m(x_i)]$$

for $1 \leq i \leq m$. Define matrices

$$X_i = (A^i)^\top A^i.$$

Denote $Y = \sum_{i=1}^m X_i$. Then $\lambda_m(\mathbb{E}Y) = m\gamma_m$ using the fact that $\mathbb{E}\left[\phi_i(x)\phi_j(x)\right] = \delta_{ij}$. Furthermore, $X_i \succeq 0$ and

$$\|X_i\| \leq \mathrm{tr}(X_i) = \sum_{u=1}^m \gamma_u \phi_u^2(x_i) \leq g(x_i, x_i) = c_g.$$

Therefore, matrix Chernoff bound (e.g., [123]) gives

$$\Pr\left[\lambda_m(Y) \leq (1-\epsilon)\lambda_m(\mathbb{E}Y)\right] \leq m \exp\left(-(1-\epsilon)^2 \lambda_m(\mathbb{E}Y)/(2c_g)\right).$$

Choose $\epsilon = 1/2$ and use the facts that $G^{[m]} = AA^\top$, $Y = A^\top A$ and $\lambda_m(G^{[m]}) = \lambda_m(Y)$, we finish the proof. ■

Proof [Proof of Lemma 28] By Weyl's theorem and the fact that E^m is PSD,

$$\lambda_m(G) \geq \lambda_m(G^{[m]}) + \lambda_m(E^m) \geq \lambda_m(G^{[m]}).$$

Lemma 28 then follows from Lemma 53. ■

C.3 Bounding the difference between $\lambda_m(G)$ and $\lambda_m(G_n)$: Proof of Lemma 29

Using Weyl's theorem we have that

$$|\lambda_m(G_n) - \lambda_m(G)| \leq \|G_n - G\|. \tag{C.9}$$

We are going to give an upper bound on $\|G_n - G\|$:

$$\|G_n - G\| = \sup_{\|\alpha\|=1} \sum_{i,j=1}^{m} \alpha_i \alpha_j (x_i^\top x_j) E_{ij}, \tag{C.10}$$

$$\text{where } E_{i,j} = \frac{1}{n} \sum_{k=1}^{n} \sigma'(w_k^\top x_i)\sigma'(w_k^\top x_j) - \mathbb{E}_w[\sigma'(w^\top x_i)\sigma'(w^\top x_j)], \tag{C.11}$$

and the first expectation is taken over w uniformly on the sphere $\mathbb{S}^{d-1}$.

Our bound heavily relies on the inner products $|\langle x_i, x_j \rangle|$ for all $i \neq j$ being small enough. In the next lemma, we provide such a result for uniformly distributed data.

Lemma 54 (Tail bound for spherical distribution) *If a and b are independent vectors uniformly distributed over the unit sphere $\mathbb{S}^{d-1}$, then there exists a constant $c > 0$, such that for any $u > 0$,*

$$\Pr\left[|\langle a, b \rangle| \geq \frac{cu}{\sqrt{d}}\right] \leq 2e^{-u^2}.$$

Proof Note that both a and b are sub-gaussian random variables with sub-gaussian norm $c/\sqrt{d}$ where c is some constant [143].

Denote $\mathbb{E}_b[\cdot]$ the expectation over b. We can rewrite the probability as

$$\begin{aligned} \Pr\left[|\langle a, b \rangle| \geq \frac{cu}{\sqrt{d}}\right] &\leq \mathbb{E}_b \Pr\left[|\langle a, b \rangle| \geq \frac{cu}{\sqrt{d}} \mid b\right] \\ &\leq \mathbb{E}_b\left\{2\exp\left(-u^2\right)\right\} = 2\exp\left(-u^2\right). \end{aligned} \tag{C.12}$$

The last inequality uses the independence of a and b and $\|\langle a, b \rangle\|_{\psi_2} \leq \|b\|_2 \|a\|_{\psi_2}$ for a fixed b. ■

Decomposing the sum into diagonal and off-diagonal terms gives us

$$\|G_n - G\| \leq \sup_{\|\alpha\|=1} \sum_{i\neq j}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle E_{ij} + \sum_{i=1}^{m} \alpha_i^2 E_{ii} \tag{C.13}$$

$$\leq \sup_{\|\alpha\|=1} \sqrt{\sum_{i\neq j} \alpha_i^2 \alpha_j^2} \sqrt{\sum_{i\neq j} \langle x_i, x_j \rangle^2 E_{ij}^2} + \max_i |E_{ii}| . \tag{C.14}$$

Let $\mathcal{G}$ denote the event that for all $i \neq j \in [m]$, $|\langle x_i, x_j \rangle| \leq O\left(\frac{\log d}{\sqrt{d}}\right)$, then by Lemma 54 and the union bound

$$\Pr[\neg\mathcal{G}] \leq 2m^2 e^{-\log^2 d}. \tag{C.15}$$

Therefore, with probability at least $1 - 2m^2 e^{-\log^2 d}$, we have

$$\|G_n - G\| \leq c\frac{\log d}{\sqrt{d}} \sqrt{\sum_{i\neq j} E_{ij}^2} + \max_i |E_{ii}| . \tag{C.16}$$

Note that

$$U(\{x_1, \cdots, x_m\}) = \frac{1}{m(m-1)} \sum_{i\neq j} E_{ij}^2 \tag{C.17}$$

is a U-statistics.

Suppose $|E_{ij}| \leq B$, according to the concentration inequality (Theorem 2 in [125]), we have with probability at least $1 - \delta$

$$\sum_{i\neq j} E_{ij}^2 \leq m(m-1)\mathbb{E}_{\{x_1,x_2\}} E_{12}^2 + m(m-1)\left(\sqrt{\frac{4\Sigma^2}{m}\log\frac{1}{\delta}} + \frac{4B^2}{3m}\log\frac{1}{\delta}\right), \tag{C.18}$$

where $\Sigma^2 = \mathbb{E}\left[E_{1,2}^4\right] - \mathbb{E}\left[E_{1,2}^2\right]^2$ is the variance for the kernel in U-statistics.

Putting everything together, we have with probability at least $1 - \delta - 2m^2 e^{-\log^2 d}$

$$\|G_n - G\| \leq c\frac{\log d}{\sqrt{d}} \left(m\sqrt{\mathbb{E}_{\{x_1,x_2\}} E_{12}^2} + m\left(\frac{4\Sigma^2}{m}\log\frac{1}{\delta}\right)^{1/4} + mB\sqrt{\frac{4}{3m}\log\frac{1}{\delta}} \right) + B \tag{C.19}$$

C.4 Discrepancy of the weights

In this section, we relate the quantities $\mathbb{E}_{\{x_1,x_2\}} E_{12}^2$ and B to the discrepancies of the weights. Note that for ReLU, $\sigma'(w^\top x)$ does not depend on the norm of w, so we can focus on w on the unit sphere.

Given a set of n points $W = \{w_i\}_{i=1}^n$ on the unit sphere $\mathbb{S}^{d-1}$, the discrepancy of W with respect to a measurable subset $S \subseteq \mathbb{S}^{d-1}$ is defined as

$$\text{dsp}(W, S) = \frac{1}{n}|W \cap S| - A(S) \tag{C.20}$$

where $A(S)$ is the normalized area of S (*e.g.*, the area of the whole sphere $A(\mathbb{S}^{d-1})$ is 1). Let $\mathcal{S}$ denote the family of slices in $\mathbb{S}^{d-1}$

$$\mathcal{S} = \left\{S_{xy} : x, y \in \mathbb{S}^{d-1}\right\}, \text{ where } S_{xy} = \left\{w \in \mathbb{S}^{d-1} : w^\top x \geq 0, w^\top y \geq 0\right\}. \tag{C.21}$$

The L_∞ discrepancy of W with respect to $\mathcal{S}$ is

$$L_\infty(W, \mathcal{S}) = \sup_{S \in \mathcal{S}} \text{dsp}(W, S), \tag{C.22}$$

and the L_2 discrepancy is

$$L_2(W, \mathcal{S}) = \sqrt{\mathbb{E}_{x,y} \text{dsp}(W, S_{xy})^2} \tag{C.23}$$

where the expectation is taken over x, y uniformly on the sphere. We use $L_\infty(W)$ and

$L_2(W)$ as their shorthands.

For ReLU, by definition, we have

$$\mathbb{E}E_{ij}^2 = \left(L_2(W)\right)^2, \tag{C.24}$$

$$B \leq L_\infty(W), \tag{C.25}$$

$$\Sigma^2 \leq \mathbb{E}\left[E_{1,2}^4\right] \leq \mathbb{E}\left[E_{1,2}^2\right] \max_{x_1,x_2} \left|E_{1,2}^2\right| \leq \left(L_\infty(W)L_2(W)\right)^2, \tag{C.26}$$

using the fact that $E_{ij} = \text{dsp}(W, S_{x_i x_j})$.

Therefore, the bound becomes

$$\|G_n - G\| \leq c\frac{\log d}{\sqrt{d}}\left(mL_2(W) + \sqrt{L_\infty(W)L_2(W)}m\left(\frac{4}{m}\log\frac{1}{\delta}\right)^{1/4} + mL_\infty(W)\sqrt{\frac{4}{3m}\log\frac{1}{\delta}}\right) + L_\infty(W) \tag{C.27}$$

In the following subsections, we will discuss the discrepancies.

C.4.1 Computing L_2 discrepancy for ReLU

Note that the derivative of ReLU $\sigma'(w^\top x) = \mathbb{I}\left[w^\top x\right]$ does not depend on the norm of w. Without loss of generality, we can assume $\|w\| = 1$ throughout this subsection.

Theorem 30 *Suppose* $W = \{w_i\}_{i=1}^n \subseteq \mathbb{S}^{d-1}$.

$$\left(L_2(W)\right)^2 = \frac{1}{n^2}\sum_{i,j=1}^n k(w_i, w_j)^2 - \mathbb{E}_{u,v}\left[k(u,v)^2\right]$$

where $\mathbb{E}_{u,v}$ *is over* u *and* v *uniformly distributed on* $\mathbb{S}^{d-1}$ *and the kernel* $k(\cdot,\cdot)$ *is*

$$k(u,v) = \frac{\pi - \arccos\langle u, v\rangle}{2\pi}.$$

Proof Let $d(u,v) = \frac{\arccos\langle u,v\rangle}{\pi}$. Let $S_{xy} = \left\{w \in \mathbb{S}^{d-1} : w^\top x \geq 0,\ w^\top y \geq 0\right\}$. It is clear

that (up to sets of measure zero)

$$A(S_{xy}) = k(x,y) = \frac{1 - d(x,y)}{2}, \tag{C.28}$$

$$\mathbb{I}\left[z \in S_{xy}\right] = \frac{1}{4}\left(\text{sign}(x^\top z) + 1\right)\left(\text{sign}(y^\top z) + 1\right), \tag{C.29}$$

where $\mathbb{I}\left[\cdot\right]$ is the indicator function. Then

$$\text{dsp}(W, S_{xy}) = \frac{1}{n}\sum_{k=1}^{n} \mathbb{I}\left[w_k \in S_{xy}\right] - A(S_{xy}) \tag{C.30}$$

$$= \frac{1}{n}\sum_{k=1}^{n} \frac{1}{4}\left(\text{sign}(x^\top w_k) + 1\right)\left(\text{sign}(y^\top w_k) + 1\right) - \frac{1 - d(x,y)}{2}. \tag{C.31}$$

Let s_{xi} be a shorthand for $\text{sign}(x^\top w_i)$. Then we have

$$\left(L_2(W)\right)^2 = \mathbb{E}_{x,y}\left(\text{dsp}(W, S_{xy})\right)^2 \tag{C.32}$$

$$= \int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \left(\frac{1}{n}\sum_{k=1}^{n} \frac{1}{4}\left(\text{sign}(x^\top w_k) + 1\right)\left(\text{sign}(y^\top w_k) + 1\right) - \frac{1 - d(x,y)}{2}\right)^2 dA(x)dA(y) \tag{C.33}$$

$$= \frac{1}{n^2}\sum_{i,j=1}^{n} \int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{(s_{xi}+1)(s_{yi}+1)}{4}\frac{(s_{xi}+1)(s_{yi}+1)}{4} dA(x)dA(y) \tag{C.34}$$

$$- \frac{2}{n}\sum_{i=1}^{n} \int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1 - d(x,y)}{2}\frac{(s_{xi}+1)(s_{yi}+1)}{4} dA(x)dA(y) \tag{C.35}$$

$$+ \int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \left(\frac{1 - d(x,y)}{2}\right)^2 dA(x)dA(y). \tag{C.36}$$

Consider the first term, which is equal to

$$\frac{1}{n^2}\sum_{i,j=1}^{n} \left(\int_{\mathbb{S}^{d-1}} \frac{(s_{xi}+1)(s_{xj}+1)}{4} dA(x)\right)\left(\int_{\mathbb{S}^{d-1}} \frac{(s_{yi}+1)(s_{yj}+1)}{4} dA(y)\right). \tag{C.37}$$

By Lemma 55,

$$\int_{\mathbb{S}^{d-1}} \frac{(s_{xi}+1)(s_{xj}+1)}{4} dA(x) = \frac{2-2d(w_i,w_j)}{4}, \tag{C.38}$$

so the first term is equal to

$$\frac{1}{n^2}\sum_{i,j=1}^{n}\left(\int_{\mathbb{S}^{d-1}} \frac{(s_{xi}+1)(s_{xj}+1)}{4} dA(x)\right)^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} k(w_i,w_j)^2. \tag{C.39}$$

Now consider the second term. Note that the summand is invariant to w_i, so it can be replaced by an arbitrary $p \in \mathbb{S}^{d-1}$. The second term is then equal to

$$\begin{aligned}
&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1-d(x,y)}{2}\frac{(\mathrm{sign}(x^\top p)+1)(\mathrm{sign}(y^\top p)+1)}{4} dA(x)dA(y) && \text{(C.40)}\\
=&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1-d(x,y)}{2}\mathbb{I}[p\in S_{xy}] dA(x)dA(y) && \text{(C.41)}\\
=&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1-d(x,y)}{2}\mathbb{I}[p\in S_{xy}] dA(x)dA(y)dA(p) && \text{(C.42)}\\
=&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1-d(x,y)}{2}\left[\int_{\mathbb{S}^{d-1}}\mathbb{I}[p\in S_{xy}] dA(p)\right] dA(x)dA(y) && \text{(C.43)}\\
=&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \frac{1-d(x,y)}{2}\frac{2-2d(x,y)}{4} dA(x)dA(y) && \text{(C.44)}\\
=&-2\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}} \left(\frac{1-d(x,y)}{2}\right)^2 dA(x)dA(y), && \text{(C.45)}
\end{aligned}$$

where the third step is by invariance to p and the fourth step is by Lemma 55. The theorem then follows. ■

Theorem 30 lets us compute $L_2(W)$ for a fixed W. The next lemma gives a concrete bound for a special case where W is uniformly distributed on the unit sphere.

Lemma 31 *There exists a constant c_g, such that for any $0 < \delta < 1$, with probability at*

least $1-\delta$ *over* $W=\{w_i\}_{i=1}^n$ *that are sampled from the unit sphere uniformly at random,*

$$(L_2(W))^2 \le c_g\left(\sqrt{\frac{\log d}{nd}\log\frac{1}{\delta}}+\frac{1}{n}\log\frac{1}{\delta}\right).$$

Proof By Theorem 30, we have

$$\begin{aligned}(L_2(W))^2 = \frac{1}{4n^2}\sum_{i,j=1}^n\left(\frac{1}{2}-d(w_i,w_j)\right)^2 - \frac{1}{4}\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}}\left(\frac{1}{2}-d(u,v)\right)^2 dA(u)dA(v)\\ + \frac{1}{4n^2}\sum_{i,j=1}^n\left(\frac{1}{2}-d(w_i,w_j)\right). \qquad \text{(C.46)}\end{aligned}$$

First consider $T_1=\frac{1}{n^2}\sum_{i,j=1}^n\left(\frac{1}{2}-d(w_i,w_j)\right)^2-\mu$ where $\mu=\int_{\mathbb{S}^{d-1}}\int_{\mathbb{S}^{d-1}}\left(\frac{1}{2}-d(u,v)\right)^2 dA(u)dA(v)$. Rewrite $T_1=\frac{1}{4n}+\frac{n-1}{n}U(W)-\mu$ where $U(W)=\frac{1}{n(n-1)}\sum_{i\neq j}\left(\frac{1}{2}-d(w_i,w_j)\right)^2$ is a U-statistics. We upper bound $U(W)$ by using Bernstein's inequality when W is uniform over the sphere.

By Taylor expansion, we have

$$\frac{1}{2}-d(u,v)=x/\pi+x^3/6\pi+O(x^5),\ \text{where } x=u^\top v.$$

Then let $\mathcal{G}$ denote the event that $|x|=|u^\top v|\le c\sqrt{\log d/d}$ for a sufficient large constant $c>0$, so that by Lemma 54, $\Pr[\neg\mathcal{G}]\le O(1/d^4)$. Then

$$\begin{aligned}\mathbb{E}\left[U(W)\right]=\mu &= \mathbb{E}\left[\left(x/\pi+x^3/6\pi+O(x^5)\right)^2\right] && \text{(C.47)}\\ &= \mathbb{E}[x^2/\pi^2+x^4/6\pi^2+O(x^6)] && \text{(C.48)}\\ &\le \mathbb{E}\left[[x^2/\pi^2+x^4/6\pi^2+O(x^6)]\mid\mathcal{G}\right]+\Pr[\neg\mathcal{G}]\max_{u,v}\left[\frac{1}{2}-d(u,v)\right]^2 && \text{(C.49)}\\ &= O\left(\frac{\log d}{d}\right), && \text{(C.50)}\end{aligned}$$

and thus

$$\text{Var}\left[U(W)\right] = \mathbb{E}\left\{\left[\left(x/\pi + x^3/6\pi + O(x^5)\right)^2 - \mu\right]^2\right\} \tag{C.51}$$

$$= \mathbb{E}\left\{\left[x^2/\pi^2 + x^4/6\pi^2 + O(x^6) - \mu\right]^2\right\} \tag{C.52}$$

$$\leq \mathbb{E}\left\{\left[x^2/\pi^2 + x^4/6\pi^2 + O(x^6) - \mu\right]^2 \mid \mathcal{G}\right\} + \Pr[\neg\mathcal{G}] \max_{u,v}\left[\left(\frac{1}{2} - d(u,v)\right)^2 - u\right]^2 \tag{C.53}$$

$$= O\left(\frac{\log^2 d}{d^2}\right). \tag{C.54}$$

Then by Berstein's inequality, we have with probability at least $1-\delta$ over the W uniformly on the sphere,

$$|T_1| \leq O\left(\frac{\log d}{d}\sqrt{\frac{1}{n}\log\frac{1}{\delta}} + \frac{1}{n}\log\frac{1}{\delta}\right). \tag{C.55}$$

A similar argument holds for $T_2 = \frac{1}{n^2}\sum_{i,j=1}^{n}\left(\frac{1}{2} - d(w_i, w_j)\right)$. Note that

$$\text{Var}\left\{\left(\frac{1}{2} - d(u,v)\right)\right\} = \mu = O\left(\frac{\log d}{d}\right). \tag{C.56}$$

We have that with probability at least $1-\delta$ over the W uniform from the sphere,

$$|T_2| \leq O\left(\sqrt{\frac{\log d}{nd}\log\frac{1}{\delta}} + \frac{1}{n}\log\frac{1}{\delta}\right). \tag{C.57}$$

This completes the proof. ■

Below are some technical lemmas that are used in the analysis.

Lemma 55

$$\int_{\mathbb{S}^{d-1}} d(x,y)dA(x) = \frac{1}{2}, \forall y \in \mathbb{S}^{d-1}, \tag{C.58}$$

$$\int_{\mathbb{S}^{d-1}} \operatorname{sign}(x^\top y)dA(x) = 0, \forall y \in \mathbb{S}^{d-1}, \tag{C.59}$$

$$\int_{\mathbb{S}^{d-1}} \operatorname{sign}(x^\top z)\operatorname{sign}(y^\top z)dA(z) = 1 - 2d(x,y), \forall x, y \in \mathbb{S}^{d-1}. \tag{C.60}$$

Proof The first two are straightforward. The third is implicit in the proof of Theorem 1.21 in [122]. ■

C.5 The spectrum of γ_m

The spectrum of the kernel matrix $g(x,y) = \left(\frac{1}{2} - \frac{\arccos\langle x,y\rangle}{2\pi}\right)\langle x,y\rangle$ is determined by the spherical decomposition coefficients.

We need γ_m to decrease slower than $O(1/\sqrt{m})$ within a reasonable range, such as $m \leq 1000000$. Although the kernel associated with ReLU decreases faster than the desired rate, we can choose from a family of such arccos kernels such that the spectrum decays slower than $1/\sqrt{m}$. Although there is no analytical solution to the spectrum, we can compute them numerically. Figure 6.1 illustrates the spectra of several arccos kernels compared to $O(1/m)$ and $O(1/\sqrt{m})$.

C.6 Rademacher complexity and final error bounds: Proof of Theorem 25 and Theorem 24

We apply the argument in [126] to our setting to get Lemma 56. Combining it with Theorem 26 leads to Theorem 25. Further combining it with Lemma 31 leads to Theorem 24.

Lemma 56 *Suppose the data are bounded:* $|y| \le Y$ *and* $\|x\|_2 \le 1$. *Let*

$$\mathcal{F} = \left\{ f(x) = \sum_{k=1}^{n} v_k \sigma(w_k^\top x) : v_k \in \{-1, +1\}, \sum_k \|w_k\|_2 \le C_W \right\}.$$

Then with probability $\ge 1 - \delta$, *for any* $f \in \mathcal{F}$,

$$\frac{1}{2}\mathbb{E}(y - f(x))^2 \le \frac{1}{2m}\sum_{l=1}^{m}(y_l - f(x_l))^2 + \frac{2(Y + C_W)C_W}{\sqrt{m}} + (Y^2 + C_W^2)\sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \tag{C.61}$$

Proof For a sample $S = ((x_1, y_1), \cdots, (x_m, y_m))$, and a loss function $\ell(y, x) = \frac{1}{2}(y - f(x))^2$, we denote $\hat{\mathbb{E}}_S[\ell]$ as the empirical average $\hat{\mathbb{E}}_S[\ell] = \frac{1}{m}\sum_{l=1}^{m}\ell(y_l, x_l)$.

Define

$$\Phi(S) = \sup_{\ell \in \mathcal{L}} \mathbb{E}[\ell] - \hat{\mathbb{E}}_S[\ell] \tag{C.62}$$

where $\mathcal{L}$ is the set of loss functions

$$\mathcal{L} = \left\{ \ell(y, x) = \frac{1}{2}(y - f(x))^2 : f \in \mathcal{F} \right\}.$$

Let S and S' be two datasets that differ by exactly one data point (x_i, y_i) and (x_i', y_i'). Then we have a bound on the difference of loss functions. Since $\|x\|_2 \le 1$ and $\sum_k \|w_k\|_2 \le C_W$, we have $|f| \le C_W$. Thus

$$|\ell(y, f(x)) - \ell(y', f(x'))| \le \frac{1}{2}\max\left\{(y - f(x))^2, (y' - f(x'))^2\right\} \le Y^2 + C_W^2. \tag{C.63}$$

This leads to an upper bound

$$\begin{aligned}\Phi(S) - \Phi(S') &\leq \sup_{\ell \in \mathcal{L}} \hat{\mathbb{E}}_S[\ell] - \hat{\mathbb{E}}_{S'}[\ell] \\ &= \sup_{\ell \in \mathcal{L}} \frac{\ell(y_i, f(x_i)) - \ell(y_i', f(x_i'))}{m} \leq \frac{Y^2 + C_W^2}{m}. \end{aligned} \tag{C.64}$$

Similarly, we can get the other side of the inequality and have $|\Phi(S) - \Phi(S')| \leq \frac{Y^2+C_W^2}{m}$.

From McDiamids' inequality, with probability at least $1 - \delta$ we get

$$\Phi(S) \leq \mathbb{E}_S \Phi(S) + (Y^2 + C_W^2)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{C.65}$$

The first term on the right-hand side can be bounded by Rademacher complexity as shown in the book Foundations of Machine Learning (3.13). In the end, we have the bound

$$\frac{1}{2}\mathbb{E}(y - f(x))^2 \leq \frac{1}{2m}\sum_{l=1}^{m}(y_l - f(x_l))^2 + 2\mathcal{R}_m(\mathcal{L}) + (Y^2 + C_W^2)\sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{C.66}$$

where $\mathcal{R}_m(\mathcal{L})$ is the Rademacher complexity of the function class $\mathcal{L}$.

We can find the Rademacher complexity by using composition rules. The Rademacher complexity of linear functions $\left\{w^\top x : \|w\|_2 \leq b_W, \|x\|_2 \leq 1\right\}$ is $b_W/\sqrt{m}$, where m is the number of data points. If a function ϕ is L-Lipschitz, then for any function class $\mathcal{H}$, we have $\mathcal{R}(\phi \circ \mathcal{H}) \leq L\mathcal{R}(\mathcal{H})$. In addition, we also have $\mathcal{R}(cH) = |c|\,\mathcal{R}(H)$ and $\mathcal{R}(\sum_k F_k) \leq \sum_k \mathcal{R}(F_k)$.

So for the function class $\mathcal{F}$ that describes a neural network, we have

$$\mathcal{R}_m(\mathcal{F}) \leq \frac{C_W}{\sqrt{m}}. \tag{C.67}$$

It is derived by using the fact that $\sigma'(\cdot)$ is 1-Lipschitz and $\sum_k \|w_k\|_2 \leq C_W$.

Finally composing on the loss function we get

$$\mathcal{R}_m(\mathcal{L}) \leq \frac{(Y + C_W)C_W}{\sqrt{m}}, \tag{C.68}$$

using the fact that the ground truth in the loss should be bounded by Y and the function bounded by C_W, thus the Lipschitz constant of the loss function is bounded by $Y + C_W$. ■

APPENDIX D

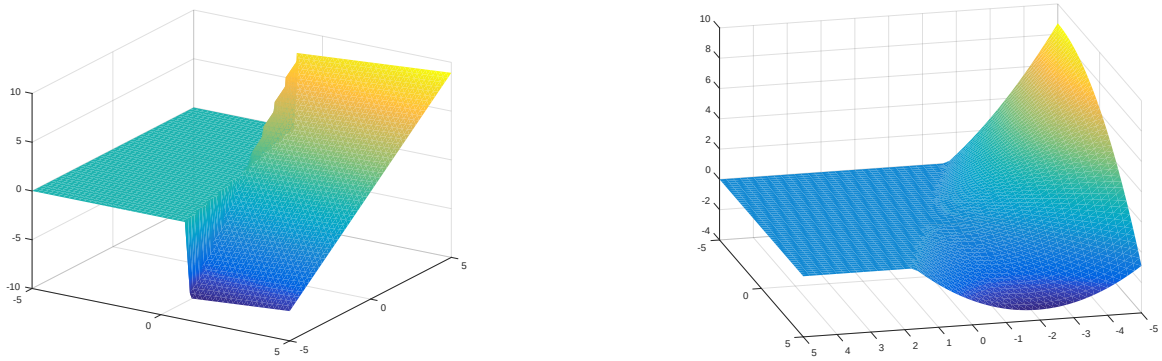
PROOFS AND ADDITIONAL EXPERIMENTS IN CHAPTER 7

D.1 Visualization of semi-random features

We visualize the surfaces of two semi-random features (a) linear (LSR) ($s = 0$) and (b) square (SSR) ($s = 1$) in Figure D.1. The semi-random features are defined by two parts. The random weights and nonlinear thresholding define a hyperplane, where on one side, it is zero and on the other side, it is either an adjustable linear or adjustable square function. During learning, gradient descent is used to tune the adjustable parts to fit target functions.

D.2 Importance of Bias Terms in First Layer

We note the importance of the bias term r_0 in the first layer, to obtain universal approximation power. Without the bias term, Theorem 32 does not hold. Indeed, it is easy to see that without the bias term, Heaviside step functions do not form a function class with universal approximation ability for $x \in \mathbb{R}^d$. The importance of the bias term can also be seen by the binomial theorem. For example, let $g(z)$ ($z \in \mathbb{R}$) be a polynomial of degree *at least* c



(a) linear semi-random

(b) square semi-random

Figure D.1: Visualization of linear semi-random (LSR) ($s = 0$) and squared semi-random (SSR) ($s = 1$) units. The random weights define a hyperplane: on one side, it is zero while on the other side, it is either adjustable linear or adjustable square function.

for some constant c, then g does *not* form a function class that contains the polynomial of degree less than c in z. However, if $z = x + r_0$, it is possible to contain polynomials of degrees less than c in x by binomial theorem.

D.3 Proofs for One Hidden Layer Model

In this section, we provide the proofs of the theorems presented in Section 7.4. We denote a constant function $x \mapsto 1$ by $\mathbb{1}$ (i.e., $\mathbb{1}(x) = 1$).

D.3.1 Proof of Theorem 32 (Universal Approximation)

To prove the theorem, we recall the following known result.

Lemma 57 [137, Proposition 1] *For any $\sigma : \mathbb{R} \to \mathbb{R}$, $\mathrm{span}\{\sigma(r^\top x + r_0) : r \in \mathbb{R}^d, r_0 \in \mathbb{R}\}$ is dense in $L^p(\Omega)$ for every compact subset $\Omega \subseteq \mathbb{R}^d$ and for every $p \in [1, \infty)$,if and only if σ is not a polynomial of finite degree (almost everywhere).*

With this lemma, we first prove that for every fixed s, the span of a set of our functions σ_s is dense in $L^2(\Omega)$.

Lemma 58 *For every fixed finite integer s and for every compact subset $\Omega \subseteq \mathbb{R}^d$, $\mathrm{span}\{\sigma_s(r^\top x + r_0), \mathbb{1}(x) : r \in \mathbb{S}^{d-1}, r_0 \in \mathbb{R}\}$ is dense in $L^2(\Omega)$, where $g(x) = 1$ represents a constant function.*

Proof Since σ_0 is Heaviside step function, σ_0 is not a polynomial of finite degree and $\mathrm{span}\{\sigma_0(r^\top x + r_0) : r \in \mathbb{R}^d, r_0 \in \mathbb{R}\}$ is dense in $L^2(\Omega)$ (we know each of both statements independently without Lemma 57). For the case of $s \geq 1$, $\sigma_s(z) = z^s \sigma_0(z)$ is not a polynomial of finite degree (i.e., otherwise, σ_0 is a polynomial of finite degree). Thus, by Lemma 57, for fixed finite integer $s \geq 1$, $\mathrm{span}\{\sigma_s(r^\top x + r_0) : r \in \mathbb{R}^d, r_0 \in \mathbb{R}\}$ is dense in $L^2(\Omega)$.

Since $\sigma_s(\frac{r^\top x}{\|r\|} + r_0) = \|r\|^{-s}\sigma_s(r^\top x + \|r\|r_0)$ for all $r \neq 0$ (because $H(z)$ is insensitive to a positive scaling of z),

$$\begin{aligned}\operatorname{span}\{\sigma_s(r^\top x + r_0) : r \in \mathbb{R}^d, r_0 \in \mathbb{R}\} &= \operatorname{span}\{\sigma_s(r^\top x + r_0) : r \in \mathbb{S}^{d-1} \cup \{0\}, r_0 \in \mathbb{R}\} \\ &= \operatorname{span}\{\sigma_s(r^\top x + r_0), \mathbb{1}(x) : r \in \mathbb{S}^{d-1}, r_0 \in \mathbb{R}\}.\end{aligned}$$

This completes the proof. ■

In practice, we want to avoid wasting samples; we want to have a choice to sample r_0 from the interval that matters. In order to do that, we admit the dependence of our function class on Ω via the following lemma.

Lemma 59 *For every fixed finite integer s and for every compact subset $\Omega \subseteq \mathbb{R}^d$,* $\operatorname{span}\{\sigma_s(r^\top x + r_0), \mathbb{1}(x) : r \in \mathbb{S}^{d-1}, |r_0| \leq \operatorname{radius}(\Omega)\}$ *is dense in $L^2(\Omega)$.*

Proof For any $r_0 \in \mathbb{R} \setminus [-\operatorname{radius}(\Omega), +\operatorname{radius}(\Omega)]$, since $|r^\top x| \leq \|r\|_2\|x\|_2 \leq \operatorname{radius}(\Omega)$, either $\sigma_s(r^\top x + r_0) = 0$ or $= (r^\top x + r_0)^s$, for all $x \in \Omega$. Meanwhile, with $r_0 = \operatorname{radius}(\Omega)$, $\sigma_s(r^\top x + r_0) = (r^\top x + r_0)^s$, for all $x \in \Omega$. Since $(r^\top x + r_0)^s = \sum_{k=0}^{s} \binom{s}{k} r_0^{s-k}(r^\top x)^k$, it is a polynomial in x, and r_0 only affects the coefficients of the polynomial. Thus, for any $x \in \Omega$, $\operatorname{span}\{\sigma_s(r^\top x + r_0), \mathbb{1}(x) : r \in \mathbb{S}^{d-1}, r_0 \in \mathbb{R}\} = \operatorname{span}\{\sigma_s(r^\top x + r_0), \mathbb{1}(x) : r \in \mathbb{S}^{d-1}, |r_0| \leq \operatorname{radius}(\Omega)\}$, because the difference in the nonzero coefficients of the polynomials can be taken care of by adjusting the coefficients of the linear combination, unless the right-hand side only contains $r_0 \in \{0\}$. Having only $r_0 \in \{0\}$ implies $\Omega = \{0\}$, which is assumed to be false. ■

We use the following lemma to translate the statement with uncountable samples in Lemma 59 to countable samples in Theorem 32.

Lemma 60 *Let s be a fixed finite integer, Ω be any compact subset of $\mathbb{R}^d$, and $(\bar{r}, \bar{r}_0)$ is in*

$$\mathbb{S}^{d-1} \times [-\text{radius}(\Omega), \text{radius}(\Omega)].$$

Then, for any $\epsilon > 0$, there exists a δ such that for any $(r, r_0) \in B((\bar{r}, \bar{r}_0), \delta)$,

$$\int_{x \in \Omega} (\sigma_s(r^\top x + r_0) - \sigma_s(\bar{r}^\top x + \bar{r}_0))^2 < \epsilon.$$

Proof For the case of $s = 0$, it directly follows the proof of lemma II.5 in [144]. For the case of $s \geq 1$, we can think of σ_s as a uniformly continuous function with a compact domain that contains all the possible inputs of σ_s for our choice of Ω and $\mathbb{S}^{d-1} \times [-\text{radius}(\Omega), \text{radius}(\Omega)]$. The lemma immediately follows from its uniform continuity. ■

Note that if $s \to \infty$, $\|x\| \to \infty$, $\|r\| \to \infty$, or $\|r_0\| \to \infty$, the proof of Lemma 60 does not work. We are now ready to prove Theorem 32.

Proof of Theorem 32 Fix s and let $\sigma_i : x \mapsto \sigma_s(r_i^\top x + r_{0i})$ with r_i and r_{0i} being sampled randomly for $i = 2, 3, \ldots$ as specified in Section 7.3. Let $\sigma_1 : x \mapsto \sigma_s(r_1^\top x + r_{01}) = \mathbb{1}(x) = 1$ as specified in Section 7.4. Since $\text{span}\{\sigma_1, \sigma_2, ..., \sigma_n\} \subseteq \mathcal{F}_n^s$ (as we can get $\text{span}\{\sigma_1, \sigma_2, ..., \sigma_n\}$ by only using $w_{0i}^{(1)}$ and $w_i^{(2)}$), we only need to prove the universality of $\text{span}\{\sigma_1, \sigma_2, ..., \sigma_n\}$ as $n \to \infty$. We prove the statement by contradiction. Suppose that $\text{span}\{\sigma_1, \sigma_2, ...\}$ is not dense in $L^2(\Omega)$ (with some nonzero probability). Then, there exists a nonzero $f_0 \in L^2(\Omega)$ such that $\langle f_0, \sigma_i \rangle = 0$ for all $i = 1, 2, \ldots$. However, from Lemma 59, either $\langle f_0, \sigma_1 \rangle \neq 0$ or there exists $(\bar{r}, \bar{r}_0) \in \mathbb{S}^{d-1} \times [-\text{radius}(\Omega), \text{radius}(\Omega)]$ such that $| \langle f_0, \sigma_{\bar{r}, \bar{r}_0} \rangle | = |c| > 0$ with some constant c, where $\sigma_{\bar{r}, \bar{r}_0} = \sigma_s(\bar{r}^\top x + \bar{r}_0)$. If $\langle f_0, \sigma_1 \rangle \neq 0$, we get the desired contradiction, and hence we considerer the latter case. In the latter case,

for any $\epsilon > 0$,

$$\begin{aligned}
|c| &= |\langle f_0, \sigma_{\bar{r},\bar{r}_0} \rangle| \\
&= |\langle f_0, \sigma_{\bar{r},\bar{r}_0} \rangle - \langle f_0, \sigma_i \rangle| \ \ \forall i \in \{2,3,...\} \\
&= |\langle f_0, \sigma_{\bar{r},\bar{r}_0} - \sigma_i \rangle| \ \ \forall i \in \{2,3,...\} \\
&\leq \|\sigma_{\bar{r},\bar{r}_0} - \sigma_i\| \|f_0\| \ \ \forall i \in \{2,3,...\} \\
&< \epsilon \|f_0\| \ \ \exists i \in \{2,3,...\},
\end{aligned}$$

where the last line follows from lemma 60 and our sampling procedure; that is, for any $\epsilon > 0$, if a sampled (r_i, r_{0i}) is in a δ-ball, $\|\sigma_i - \sigma_{\bar{r},\bar{r}_0}\| < \epsilon$ (from Lemma 60). Since our sampling procedure allocates nonzero probability on any such interval, as $i \to \infty$, the probability of sampling (r_i, r_{0i}) in any δ-ball becomes one. This shows the last line. The above inequality leads a contradiction $|c| < |c|$ by choosing $\epsilon < |c|/\|f_0\|$. This completes the proof. ■

D.3.2 Proof of Theorem 33 (No Bad Local Minima and Few Bad Critical Points)

Since multiplying a constant in w does not change the optimization problem (in terms of optimizer), in the proof of Theorem 33, we consider

$$\mathcal{L}(w) := \sum_{i=1}^{m} \left(y_i - \hat{f}_n^s(x_i; w) \right)^2,$$

to be succinct.

Proof of Theorem 33 (iii) and (iv) Define a $m \times (nd)$ matrix as

$$D' = \begin{bmatrix} w_1^{(2)}\sigma_s(\mathbf{x}_1^\top \mathbf{r}_1)\mathbf{x}_1^\top & \cdots & w_n^{(2)}\sigma_s(\mathbf{x}_1^\top \mathbf{r}_n)\mathbf{x}_1^\top \\ \vdots & \ddots & \vdots \\ w_1^{(2)}\sigma_s(\mathbf{x}_m^\top \mathbf{r}_1)\mathbf{x}_m^\top & \cdots & w_n^{(2)}\sigma_s(\mathbf{x}_m^\top \mathbf{r}_n)\mathbf{x}_m^\top \end{bmatrix}.$$

where $w_j^{(2)}\sigma_s(\mathbf{x}_i^\top \mathbf{r}_j)\mathbf{x}_i^\top$ is a $1 \times d$ block at (i,j)-th block entry. Then, we can rewrite the objective function as

$$\mathcal{L}(w) = \|D'\mathrm{vec}(w^{(1)}) - Y\|_2^2,$$

where

$$\mathrm{vec}(w^{(1)}) = (w_1^{(1)\top}, w_2^{(1)\top}, \ldots, w_n^{(1)\top})^\top.$$

Therefore, at any critical point w.r.t. $w^{(1)}$ (i.e., by taking gradient of $\mathcal{L}$ w.r.t. $w^{(1)}$ and setting it to zero), $\hat{Y}$ is the projection of Y onto the column space of D'.

Since $\hat{f}_n(x_i; w) = \sum_{k=1}^n \sigma_s(\mathbf{x}_i^\top \mathbf{r}_k)(\mathbf{x}_i^\top \mathbf{w}_k^{(1)})w_k^{(2)}$,

$$\mathcal{L}(w) = \|D[[w]] - Y\|_2^2,$$

where

$$[[w]] = (w_1^{(2)}w_1^{(1)\top}, w_2^{(2)}w_2^{(1)\top}, \ldots, w_n^{(2)}w_n^{(1)\top})^\top. \tag{D.1}$$

Therefore, we achieve a global minimum if $\hat{Y}$ is the projection of Y onto the column space of D.

If $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$, the column space of D' is equal to that of D. Hence, $\hat{Y}$ being the projection of Y onto the column space of D is achieved by our parameterization, which completes the proof of Theorem 33 (iv). Any critical point w.r.t. $(w^{(1)}, w^{(2)})$ needs to be a critical point w.r.t. $w^{(1)}$. Hence, every critical point is a global minimum if $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$. This completes the proof of Theorem 33

(iii). ■

Proof of Theorem 33 (ii) From Theorem 33 (iii), if $w_k^{(2)} \neq 0$ for all $k \in \{1, 2, \ldots, n\}$, every local minimum is a global minimum (because a set of local minima is included in a set of critical points).

Consider any point $\bar{w}$ where $\bar{w}_k^{(2)} = 0$ for some $k \in \{1, 2, \ldots, n\}$. Without loss of generality, with some integer c, let $\bar{w}_k^{(2)} \neq 0$ for all $k \in \{1, \ldots, c-1\}$ and $\bar{w}_k^{(2)} = 0$ for all $k \in \{c, \ldots, n\}$.

Then, at the point $\bar{w}$,

$$\mathcal{L}(\bar{w}) = \|D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y\|_2^2,$$

where $D_{\cdot 1:c-1}$ and $[[\bar{w}]]_{1:c-1}$ are the block elements of D (equation 7.4) and $[[\bar{w}]]$ (equation D.1) that correspond to the nonzero $\bar{w}_k$. Then, from the proof of Theorem 33 (iii), if $\bar{w}$ is a critical point, then $D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1}$ is the projection of Y onto $D_{\cdot 1:c-1}$.

If $\bar{w}$ is a local minimum, for sufficiently small $\epsilon > 0$, for any $\bar{k} \in \{c, \ldots, n\}$, for any $(w_{\bar{k}}^{(1)}, w_{\bar{k}}^{(2)}) \in \mathbb{R}^d \times \mathbb{R}$,

$$\begin{aligned} &\|D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y\|_2^2 \\ &\leq \|D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} + D_{\cdot \bar{k}}[[\bar{w}_\epsilon]]_{\bar{k}} - Y\|_2^2, \end{aligned}$$

which is simplified to

$$\begin{aligned} 0 \leq & 2(D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y)^\top (D_{\cdot \bar{k}}[[\bar{w}_\epsilon]]_{\bar{k}}) \\ & + \|D_{\cdot \bar{k}}[[\bar{w}_\epsilon]]_{\bar{k}}\|^2, \end{aligned} \tag{D.2}$$

where $[[\bar{w}_\epsilon]]_{\bar{k}}$ is a ϵ-perturbation of $[[\bar{w}]]_{\bar{k}}$ as

$$[[\bar{w}_\epsilon]]_{\bar{k}} = \epsilon w_{\bar{k}}^{(2)}(\bar{w}_{\bar{k}}^{(1)} + \epsilon w_{\bar{k}}^{(1)}).$$

Here, where $D_{\cdot,\bar{k}}$ is the corresponding $\bar{k}$-th block of size $m \times d$. In equation (D.2), the first term contains ϵ and ϵ^2 terms whereas the second term contains ϵ^2, ϵ^3 and ϵ^4 terms. With ϵ sufficiently small, the ϵ term must be zero (as we can change its sign by the direction of perturbation). Then, with ϵ sufficiently small, the ϵ^2 terms become dominant and must satisfy

$$\begin{aligned} 0 \leq & 2\epsilon^2 (D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y)^\top (D_{\cdot\bar{k}} w_{\bar{k}}^{(1)} w_{\bar{k}}^{(2)}) \\ & + \epsilon^2 \|D_{\cdot\bar{k}} \bar{w}_{\bar{k}}^{(1)} w_{\bar{k}}^{(2)}\|^2, \end{aligned}$$

for any $(w_{\bar{k}}^{(1)}, w_{\bar{k}}^{(2)}) \in \mathbb{R}^d \times \mathbb{R}$, which implies

$$0 \leq 2\epsilon^2 (D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y)^\top (D_{\cdot\bar{k}} w_{\bar{k}}^{(1)} w_{\bar{k}}^{(2)}),$$

for any $|w_{\bar{k}}^{(2)}| << \min_{j\in\{1,2,\ldots,d\}} |(w_{\bar{k}}^{(1)})_j|$ (see footnote[1]). It implies that for any $\bar{k} \in \{c, \ldots, n\}$,

$$0 = (D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y)^\top D_{\cdot\bar{k}}.$$

Since $D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1}$ is the projection of Y onto $D_{\cdot 1:c-1}$ (as discussed above), this implies that $D^\top(D[[\bar{w}]] - Y) = (D_{\cdot 1:c-1}[[\bar{w}]]_{1:c-1} - Y)^\top D = 0$. Thus, $D[[\bar{w}]]$ is the projection of Y onto D, which is a global minimum. ■

Proof of Theorem 33 (i) It is sufficient to prove non-convexity w.r.t. the parameters

[1]For a reader who have experienced deriving some conditions of a local minimum, it may sound too good to be true that we can ignore the ϵ^2 term from $\|D_{\cdot\bar{k}}[[\bar{w}_\epsilon]]_{\bar{k}}\|^2$. However, this is true and not that good since we are considering a spacial case of $\bar{w}_{\bar{k}}^{(2)} = 0$. Indeed, if $\bar{w}_{\bar{k}}^{(2)} \neq 0$, $[[\bar{w}_\epsilon]]_{\bar{k}} = (\bar{w}_{\bar{k}}^{(2)} + \epsilon w_{\bar{k}}^{(2)})(\bar{w}_{\bar{k}}^{(1)} + \epsilon w_{\bar{k}}^{(1)})$ creates the ϵ^2 term from $\|D_{\cdot\bar{k}}[[\bar{w}_\epsilon]]_{\bar{k}}\|^2$ that we cannot ignore.

$(w_k^{(1)}, w_k^{(2)})$ for each $k \in \{1, 2, \ldots, n\}$; that is, we prove that $\mathcal{L}'(w_1^{(k)}, w_k^{(2)}) = \mathcal{L}(w)|_{w_{i\neq k}=0} = \|D_{\cdot k} w_k^{(1)} w_k^{(2)} - Y\|_2^2$ is non-convex, where $D_{\cdot,k}$ is the corresponding k-th block of size $m \times d$. It is indeed easy to see that $\mathcal{L}'$ is non-convex. For example, at $(w_k^{(1)}, w_k^{(2)}) = 0$, its Hessian is

$$\begin{bmatrix} 0 & D_{\cdot k} \\ D_{\cdot k} & 0 \end{bmatrix} \not\succeq 0,$$

if $D_{\cdot k} \neq 0$. Thus, it is non-convex if $D \neq 0$. ■

D.3.3 Proof of Theorem 34 (Generalization Bound for Shallow Model)

With a standard use of Rademacher complexity (e.g., see section 3 in [138] for a clear introduction of Rademacher complexity, and the proof of lemma 12 in [136] for its concrete use),

$$\begin{aligned} &\frac{1}{2}\mathbb{E}_x(f(x) - \hat{f}(x; w^*))^2 - \mathcal{L}(w) \\ &\leq (C_Y^2 + C_{\hat{Y}}^2)\sqrt{\frac{\log \frac{1}{\delta}}{2m}} + 2(C_Y + C_{\hat{Y}})\mathcal{R}_m(\mathcal{F}), \end{aligned} \tag{D.3}$$

with probability at least $1 - \delta$, where $(C_Y^2 + C_{\hat{Y}}^2)$ is an upper bound on the value of $\mathcal{L}(w)$, and $(C_Y + C_{\hat{Y}})$ comes from an upper bound on the Lipschitz constant of the squared loss function.

We compute an upper bound on Rademacher complexity of our model class $\mathcal{F}$ as fol-

lows: with Rademacher variables ξ_i,

$$
\begin{aligned}
& m\mathcal{R}_m(\mathcal{F}) \\
&= \mathbb{E}\left[\sup_{w\in S_w}\sum_{i=1}^{m}\xi_i \mathbf{x}_i^\top \mathbf{W}^{(1)}\operatorname{diag}(\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i))W^{(2)}\right] \\
&= \mathbb{E}\left[\sup_{w\in S_w}\left(\sum_{i=1}^{m}\xi_i \mathbf{x}_i^\top \mathbf{W}^{(1)}\operatorname{diag}(\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i))\right)W^{(2)}\right] \\
&\leq \mathbb{E}\left[\sup_{w\in S_w}\left\|\sum_{i=1}^{m}\xi_i \mathbf{x}_i^\top \mathbf{W}^{(1)}\operatorname{diag}(\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i))\right\|_2 \left\|W^{(2)}\right\|_2\right] \\
&\leq \mathbb{E}\left[\sup_{w\in S_w}\left\|\sum_{i=1}^{m}\xi_i \|\mathbf{x}_i\|_2 \|\mathbf{W}^{(1)}\|_2 \|\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i)\|_2\right\| \left\|W^{(2)}\right\|_2\right] \\
&= \sup_{w\in S_w}\|\mathbf{W}^{(1)}\|_2 \left\|W^{(2)}\right\|_2 \mathbb{E}\left[\left\|\sum_{i=1}^{m}\xi_i \|\mathbf{x}_i\|_2 \|\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i)\|_2\right\|\right]
\end{aligned}
$$

where the third follows the Cauchy-Schwarz inequality and the forth line follows the properties of operator norm, $\|Ax\|_2 \leq \|A\|_2\|x\|_2$ and $\|A\|_2 \leq \|A\|_F$. Furthermore,

$$
\begin{aligned}
& \mathbb{E}\left[\left\|\sum_{i=1}^{m}\xi_i \|\mathbf{x}_i\|_2 \|\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i)\|_2\right\|\right] \\
&\leq \sqrt{\mathbb{E}\left[\left(\sum_{i=1}^{m}\xi_i \|\mathbf{x}_i\|_2 \|\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i)\|_2\right)^2\right]} \\
&\leq \sqrt{\sum_{i=1}^{m}\|\mathbf{x}_i\|_2^2 \|\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i)\|_2^2} \\
&\leq C_X C_\sigma \sqrt{m}.
\end{aligned}
$$

where the first line uses Jensen's inequality for the concave function, the third line follows that for any z,

$$
\mathbb{E}\left[\sum_{j=1}^{m}\sum_{i=1}^{m}\xi_i\xi_j z_i z_j\right] \leq \sum_{i=1}^{m} z_i^2,
$$

where we used the definition of Rademacher variables for $E[\xi_i\xi_j]$. Hence, $\mathcal{R}_m(\mathcal{F}) \leq$

$\frac{C_X C_\sigma C_{W^{(1)}} C_{W^{(2)}}}{\sqrt{m}} = \frac{C_{\hat{Y}}}{\sqrt{m}}$.

■

D.4 Proofs for Multilayer Model

In this section, we provide the proofs of the theorems and corollary presented in Section 7.5.

D.4.1 Proof of Corollary 35 (Universal Approximation with Deep Model)

From the definition of $\mathbf{r}_1^{(1)}, r_1^{(2)}, r_1^{(3)}, \ldots, r_1^{(H)}$, the first unit in each hidden layer of the main semi-random net is not affected by the random net; the output of the first unit of semi-random net is multiplied by one. By setting most of the weights w to zeros, we can use only the first unit at each layer from the second hidden layer, creating a single path that is not turned-off by the random activation net from the second hidden layer. Then, by adjusting weights w in the path, we can create an identity map from the second hidden layer unit, from which the path starts. In theorem 32, we have already shown that the output of any second hidden layer unit has universal approximation ability. This completes the proof.

D.4.2 Proof of Theorem 36 (Lower Bound on Universal Approximation Power)

Since the output of our network is the sum of the outputs of all the paths in the network, we observed in Section 7.5.1 that

$$\begin{aligned}&\hat{f}_{n_1,\ldots,n_H}(x)\\&= \sum_{k_0=0}^{d}\sum_{k_1=1}^{n_1}\cdots\sum_{k_H=1}^{n_H}[[\sigma]]_{k_1,\ldots,k_H}(x)\mathbf{x}_{k_0}[[w]]_{k_0,k_1,\ldots,k_H},\end{aligned}$$

where

$$[[\sigma]]_{k_1,\ldots,k_H}(x) = \sigma_s(\mathbf{x}^\top \mathbf{r}_{k_1}^{(1)})\prod_{l=2}^{H}\sigma_s(h_r^{(l-1)}(x)r_{k_l}^{(l)}),$$

and

$$[[w]]_{k_0,k_1,\ldots,k_H} = \left(\prod_{l=1}^{H} w^{(l)}_{k_{l-1}k_l}\right) w^{(H+1)}_{k_H}.$$

By re-writing $g_{k0,k1,\ldots,k_H}(x) = [[\sigma]]_{k_1,\ldots,k_H}(x)\mathbf{x}_{k_0}$, this means that

$$\mathcal{F}^{s}_{n_1,\ldots,n_H} \subseteq \mathrm{span}(\{g_{k0,k1,\ldots,k_H} : \forall k_0, \forall k_1, \ldots, \forall k_H\}).$$

Then, Theorem 36 is a direct application of the following result:

Lemma 61 [109, Theorem 6] *For any fixed set of M functions $g_1, g_2, \ldots, g_M$,*

$$\sup_{f\in\Gamma_C} \inf_{\hat{f}\in\mathrm{span}(g_1,g_2,\ldots,d_M)} \|f - \hat{f}\|_{L^2(\Omega)} \geq \kappa \frac{C}{d} M^{-1/d}.$$

We apply Lemma 61 with our $g_{k0,k1,\ldots,k_H}$ to obtain the statement of Theorem 36.

■

D.4.3 Proof of Corollary 37 (No Bad Local Minima and Few Bad Critical Points w.r.t. Two Last Layers)

It directly follows the proof of Theorem 33, because any critical point (or any local minimum) is a critical point (or respectively, a local minimum) with respect to $(W^{(H)}, W^{(H+1)})$.

■

D.4.4 Proof of Corollary 38 (Generalization Bound for Deep Model)

It directly follows the proof of Theorem 34 (Generalization Bound – shallow); in the proof of Theorem 34, replace $W^{(1)}\,\mathrm{diag}(\sigma_s(\mathbf{x}_i^\top \mathbf{R}_i))$ by $\prod_{l=1}^{H} W^{(l)}\,\mathrm{diag}(h_r^{(l)}(x_i))$ and bound the whole term by a product of operator norms, as it is done in the proof of Theorem 34.

■

D.5 Discussion on an Upper Bound on Approximation Error for Multilayer Model

Here, we continue the discussion in Section 7.5.2. An upper bound on approximation error can be decomposed into three illustrative terms as

$$\begin{aligned}
&\|f - \hat{f}\|^2_{L^2(\Omega)} \\
&= \sum_{i \in S} \|f - \hat{f}\|^2_{L^2(\Omega_i)} \\
&\leq \sum_{i \in S} \|f - f_i\|^2_{L^2(\Omega_i)} + \|\hat{f} - \hat{f}_i\|_{L^2(\Omega_i)} + \|f_i - \hat{f}_i\|^2_{L^2(\Omega_i)},
\end{aligned} \tag{D.4}$$

where $\{\Omega_i\}_{i \in S}$ is an arbitrary partition of Ω, and $f_i = f(x_i)$ is a constant function with some $x_i \in \Omega_i$. Then, the first term and the second term represent how much each of f and $\hat{f}$ varies in Ω_i. In other words, we can bound these two terms by some "smoothness" of f and $\hat{f}$ (some discontinuity at a set of measure zero poses no problem as we are taking the norm of $L^2(\Omega_i)$ space). The last term in equation (D.4) represents the expressive power of $\hat{f}$ on the finite points $\{x_i\}_{i \in S}$ (assuming that S is finite). Essentially, to obtain a good upper bound, we want to have a $\hat{f}$ that is "smooth" and yet expressive on some finite points.

For the first term in equation (D.4), we can obtain a bound via a simple calculation for any $f \in \Gamma_C$ as follows: by the mean value theorem, there exists z_x for each x such that

$$\begin{aligned}
\|f - f_i\|^2_{L^2(\Omega_i)} &= \int_{x \in \Omega_i} (f(x) - f(x_i))^2 \\
&= \int_{x \in \Omega_i} (\nabla f(z_x)^\top (x - x_i))^2 dx \\
&\leq \|\nabla f(z_x)\|^2_{L^2(\Omega_i)} \|x - x_i\|^2_{L^2(\Omega_i)} \\
&\leq \|\nabla f(x)\|^2_{L^2(R^d)} \|x - x_i\|^2_{L^2(\Omega_i)} \\
&\leq C^2 \|x - x_i\|^2_{L^2(\Omega_i)}.
\end{aligned}$$

where $\|x - x_i\|^2_{L^2(\Omega_i)} = \int_{x \in \Omega_i} \|x - x_i\|_2^2$. Hence, the bound goes to zero as the size of Ω_i decreases.

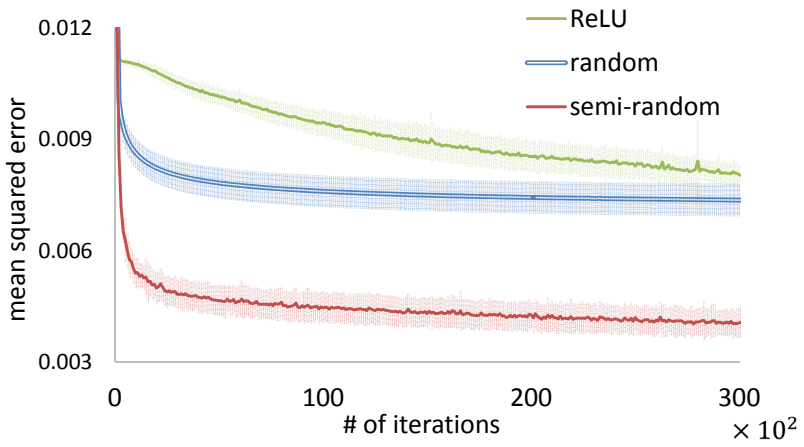


Figure D.2: Training error for a simple test function.

We can use the same reasoning to bound the second term in equation (D.4) with some "smoothness" of $\hat{f}$. A possible concern is that $\hat{f}$ becomes less "smooth" as the width n and depth H increase. However, from Bessel's inequality with Gram-Schmidt process, it is clear that such an effect is bounded for a best $\hat{f}$ with a finite $s \geq 1$ (notice the difference from statistical learning theory, where we cannot focus on the best $\hat{f}$). The last term in equation (D.4) represents the error at points $x_i \in \Omega_i$, which would be bounded via optimization theory, as the expressive power of $\hat{f}$ increases as depth H and width increase.

While this reasoning illustrates what factors may matter, a formal proof is left to future work.

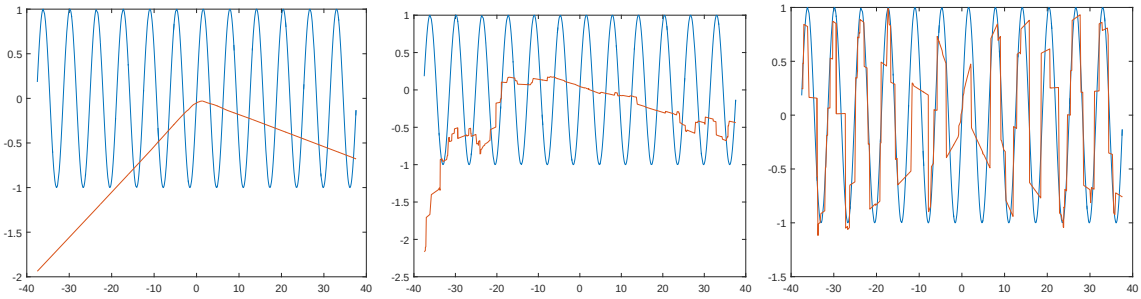
D.6 Additional Experimental Details

In this section, we provide additional Experimental details.

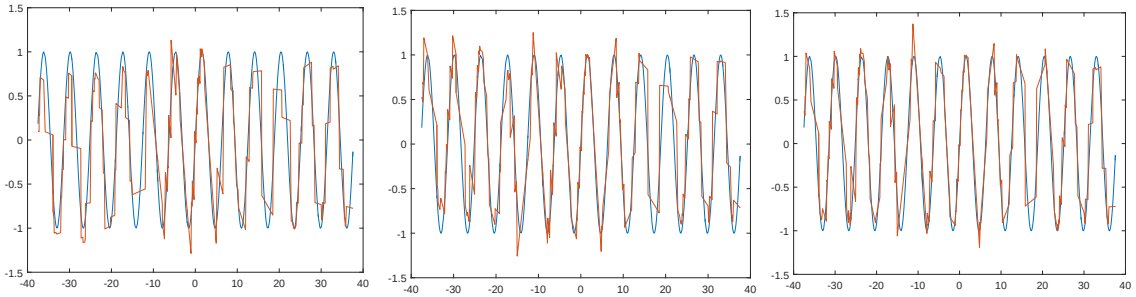
D.6.1 A simple test function

Figure D.2 shows the training errors for a sine function experiment discussed in Section 7.6.1. It shows roughly the same patterns as in their test errors, indicating that there is no large degree of overfitting.

Figures D.3, D.4, and D.5 visualize the function learned at each iteration for each

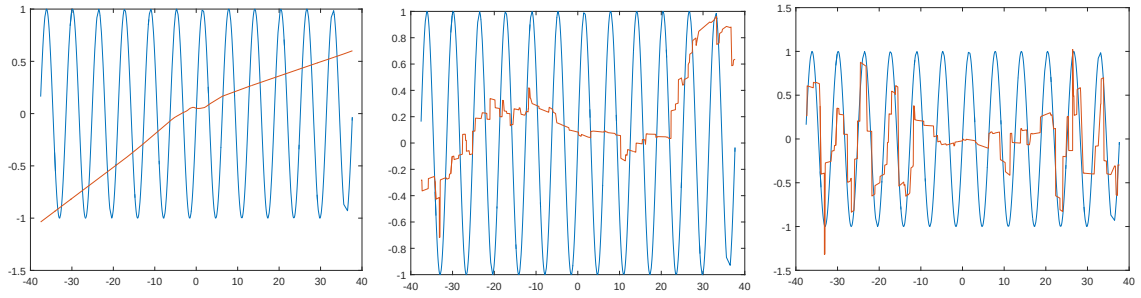


(a) iteration 0 (b) iteration 10^2 (c) iteration 10^3

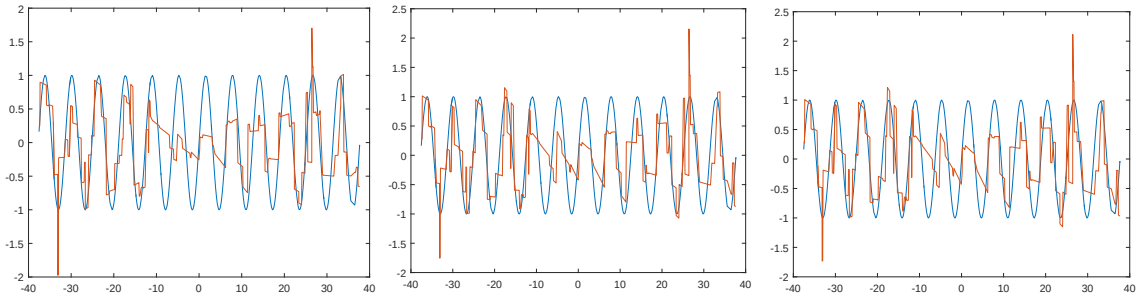


(d) iteration 10^4 (e) iteration 5×10^4 (f) iteration 10^5

Figure D.3: Function learned at each iteration (Semirandom - LSR).

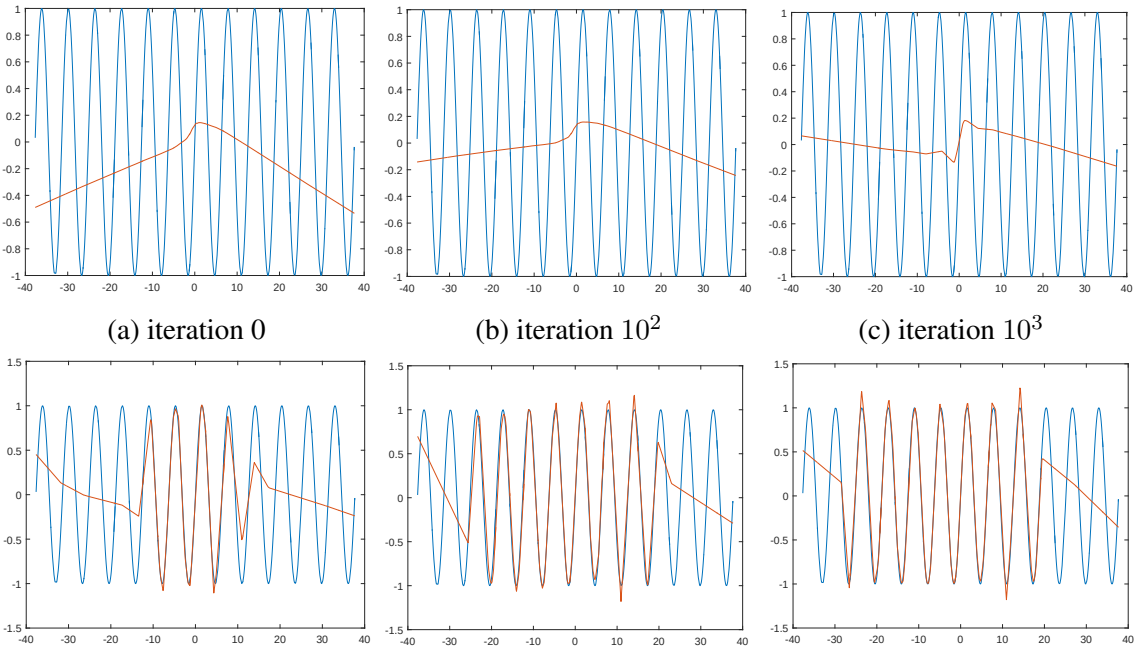


(a) iteration 0 (b) iteration 10^2 (c) iteration 10^3



(d) iteration 10^4 (e) iteration 5×10^4 (f) iteration 10^5

Figure D.4: Function learned at each iteration (fully random).



(a) iteration 0 (b) iteration 10^2 (c) iteration 10^3

(d) iteration 10^4 (e) iteration 5×10^4 (f) iteration 10^5

Figure D.5: Function learned at each iteration (ReLU).

method. We can see that semi-random features (LSR) learns the function quickly.

At each trial, 5000 points of the inputs x were sampled uniformly from $[-12\pi, 12\pi]$ for each of training dataset and test dataset. We repeated this trial 20 times to obtain standard deviations. All hyperparameters were fixed to the same values for all methods, with the network architecture being fixed to 1-50-50-1. For all the methods, we used the learning rate 5×10^{-4}, momentum parameter 0.9, and mini-batch size of 500. For all methods, the weights are initialized as 0.1 times random samples drawn from the standard normal distribution.

D.6.2 UCI datasets

For all methods on all datasets, we use SGD with 0.9 momentum, a batch size of 128 to run 100 epochs (passes over the whole dataset). The initial learning rate is set to 0.1 (for some combinations, this leads to NaN and we use 0.02 as initial learning rate). We also use an exponential staircase decaying schedule for the learning rate where after one epoch we

decrease it to 0.95 of the original rate. The parameters are initialized as normal distributions times $1/\sqrt{d}$ where d is the input dimension for the layer.

For ReLU and semi-random features (LSR and SSR), their behaviors change similarly on most datasets: more layers and more units lead to better performance. Also, on most datasets, adding more layers leads to better performance than adding more units per layer. The dataset adult is an exception where all methods have similar performance and more parameters actually lead to worse performance. This is likely due to that adult is quite noisy and using more parameters leads to overfitting. Overfitting is also observed on senseit dataset for LSR where the training errors keep become smaller but test errors increase.

D.6.3 Image datasets

We train these convolution neural networks using SGD with 0.9 momentum with a batch size of 64 for MNIST and 128 for CIFAR10 and SVHN. On each dataset, we have tried initial learning rates of 0.1, 0.01, and 0.001 and used an exponential decaying schedule. For MNIST, the decay schedule is decreasing to 0.95 of the previous rate after each epoch. For CIFAR10 and SVHN, the schedule is decreasing to 0.1 of the previous rate after 120 epochs. The best performing results on a validation set are then picked as the final solution, and usually the learning rate of 0.1 worked best. The parameters are also randomly initialized from normal distributions times $1/\sqrt{d}$ where d is the input dimension for the layer.

D.7 Proof in Section 7.7

We provide the proofs of the corollary presented in Section 7.7.

D.7.1 Proof of Corollary 39 (Lower Bound on Approximation Power for Random Feature)

The model class of any random features is the span of the features represented at the last hidden layer. Hence, the statement of the corollary directly follows from the proof of

Theorem 36.

D.8 Derivation of Upper Bounds on Approximation Errors

With an additional assumption, we compare upper bounds on approximation errors for random features and semi-random features. Recall the additional assumption that we can represent a target function f using some basis as

$$f(x) = \int_{r \in \mathbb{S}^{d-1}, \|w\| \leq C_W} \sigma(r^\top x)(w^\top x)\, p(r, w),$$

where we can write $p(r, w) = p(r)\, p(w|r)$. If we have access to the true distribution $p(r, w)$, then $f(x)$ can be approximated as a finite sample average

$$\widehat{f}_1(x) = \frac{1}{n} \sum_{i=1}^{n} \sigma(r_i^\top x)(w_i^\top x)$$
$$(r_i, w_i) \overset{\text{i.i.d.}}{\sim} p(r, w).$$

Such an approximation will incur an error of $O(\frac{1}{\sqrt{n}})$.

On the other hand, without knowing the true distribution $p(r, w)$, a purely random feature approximation of $f(x)$ will be sampling both r and w from uniform distribution: with $r_i \overset{\text{i.i.d.}}{\sim} q(r) := \text{Uniform}\left(\mathbb{S}^{d-1}\right)$ and $w_i \overset{\text{i.i.d.}}{\sim} q(w) := \text{Uniform}\left(\|w_i\| \leq C_W\right)$,

$$\begin{aligned}\widehat{f}_2(x) &= \frac{1}{n} \sum_{i=1}^{n} \frac{p(r_i, w_i)}{q(r_i) q(w_i)} \sigma(r_i^\top x)(w_i^\top x) \\ &= \frac{1}{n} \sum_{i=1}^{n} \alpha_i\, \sigma(r_i^\top x)(w_i^\top x)\end{aligned}$$

where $q(r) = q_0 := \Gamma(\frac{n}{2})/(2\pi^{\frac{d}{2}})$ (inverse of the hyper-surface area of a unit hypersphere) and $q(w) = q_1 := \Gamma(\frac{d}{2} + 1)/(\pi^{\frac{d}{2}} C_W^d)$ (inverse of the volume of a ball of radius C_W). Interestingly enough, for the unit hypersphere, the hyper-surface area reaches a maximum

and then decreases towards 0 as d increases. One can show that the seven-dimensional unit hypersphere has a maximum hyper-surface area that is less than 35. But the volume of a ball of radius C_W depends exponentially on the radius C_W. If $\|p(r,w)\|_\infty = c$, then the importance weight α_i can be as large as $\frac{c}{q_0 \cdot q_1}$. Due to the fact that q_1 can be very small which make the bound very large, this incurs a big approximation error of $O(\frac{c}{q_0 \cdot q_1 \cdot \sqrt{n}})$.

In contrast, if we sample r from unit sphere and then optimize over w (i.e., semi-random approach with one hidden layer model), we obtain the following approximation: with $r_i \overset{i.i.d.}{\sim} q(r) := \text{Uniform}\left(\mathbb{S}^{d-1}\right)$,

$$\begin{aligned}\widehat{f}_3(x) &= \frac{1}{n}\sum_{i=1}^{n} \frac{p(r_i)}{q(r_i)} \sigma(r_i^\top x)(w_i^\top x) \\ &= \frac{1}{n}\sum_{i=1}^{n} \beta_i \, \sigma(r_i^\top x)(w_i^\top x).\end{aligned}$$

By finding the best w, we can get at least as good as the case of $w_i \overset{i.i.d.}{\sim} p(w|r)$ where $p(w|r) = p(w,r)/p(r)$ is the true conditional distribution given r. If $\|p(r)\|_\infty = c$, then the important weight β_i can be much smaller, with a bound of $\frac{c}{q_0} \leq 35c$ independent of the dimension. Accordingly, this approximation incurs an error of $O(\frac{35c}{\sqrt{n}})$.

REFERENCES

[1] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[2] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *Ieee assp magazine*, vol. 3, no. 1, pp. 4–16, Jan. 1986.

[3] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *Journal of machine learning research*, vol. 3, pp. 993–1022, Jan. 2003.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society b*, vol. 39, no. 1, pp. 1–22, 1977.

[5] D. Hsu, S. Kakade, and T. Zhang, "A spectral algorithm for learning hidden markov models," in *Proc. annual conf. computational learning theory*, 2009.

[6] L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. J. Smola, "Hilbert space embeddings of hidden markov models," in *International conference on machine learning*, 2010.

[7] A. Parikh, L. Song, and E. P. Xing, "A spectral algorithm for latent tree graphical models," in *Proceedings of the international conference on machine learning*, 2011.

[8] L. Song, A. Parikh, and E. Xing, "Kernel embeddings of latent tree graphical models," in *Advances in neural information processing systems*, vol. 25, 2011.

[9] D. Foster, J. Rodu, and L. Ungar, "Spectral dimensionality reduction for hmms," *Arxiv preprint arxiv:1203.6130*, 2012.

[10] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *Arxiv preprint arxiv:1210.7559*, 2012.

[11] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu, "Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation," *Corr*, vol. abs/1204.6703, 2012.

[12] D. Hsu and S. M. Kakade, "Learning mixtures of spherical gaussians: Moment methods and spectral decompositions," in *Proceedings of the 4th conference on innovations in theoretical computer science*, ser. ITCS '13, Berkeley, California, USA: ACM, 2013, pp. 11–20, ISBN: 978-1-4503-1859-4.

[13] L. Song and B. Dai, "Robust low rank kernel embedding of multivariate distributions," in *Neural information processing systems (nips)*, 2013.

[14] E. Sgouritsa, D. Janzing, J. Peters, and B. Schölkopf, "Identifying finite mixtures of nonparametric product distributions and causal inference of confounders," in *Conference on uncertainty on artificial intelligence (uai)*, 2013.

[15] T. Benaglia, D. Chauveau, and D. R. Hunter, "An em-like algorithm for semi-and nonparametric estimation in multivariate mixtures," *Journal of computational and graphical statistics*, vol. 18, no. 2, pp. 505–526, 2009.

[16] B. Schölkopf, K. Tsuda, and J.-P. Vert, *Kernel methods in computational biology*. Cambridge, MA: MIT Press, 2004.

[17] A. J. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Proceedings of the international conference on algorithmic learning theory*, vol. 4754, Springer, 2007, pp. 13–31.

[18] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf, "Injective Hilbert space embeddings of probability measures," in *Proc. annual conf. computational learning theory*, 2008, pp. 111–122.

[19] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *Siam review*, vol. 51, no. 3, pp. 455–500, 2009.

[20] L. Wasserman, *All of nonparametric statistics*. Springer, 2006.

[21] E. Allman, C. Matias, and J. Rhodes, "Identifiability of parameters in latent structure models with many observed variables," *The annals of statistics*, vol. 37, no. 6A, pp. 3099–3132, 2009.

[22] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear algebra and its applications*, vol. 18, no. 2, pp. 95–138, 1977.

[23] L. De Lathauwer, J. Castaing, and J.-F. Cardoso, "Fourth-order cumulant-based blind identification of underdetermined mixtures," *Ieee transactions on signal processing*, vol. 55, no. 6, pp. 2965–2973, 2007.

[24] A. Anandkumar, D. J. Hsu, M. Janzamin, and S. M. Kakade, "When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity," in *Advances in neural information processing systems*, 2013, pp. 1986–1994.

[25] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade, "A tensor spectral approach to learning mixed membership community models," in *Proc. annual conf. computational learning theory*, 2013.

[26] H. Kasahara and K. Shimotsu, "Nonparametric identification of multivariate mixtures," *Journal of the royal statistical society - series b*, 2010.

[27] N. Aghaeepour, G. Finak, T. F. Consortium, T. D. Consortium, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann, "Critical assessment of automated flow cytometry data analysis techniques," *Nature methods*, vol. 10, no. 3, pp. 228–238, 2013.

[28] N. J. Proudfoot, "Ending the message: Poly (a) signals then and now," *Genes & development*, vol. 25, no. 17, pp. 1770–1782, 2011.

[29] A. A. Salamov and V. V. Solovyev, "Recognition of 3'-processing sites of human mrna precursors.," *Comput appl biosci*, vol. 13, no. 1, pp. 23–28, 1997.

[30] J. E. Tabaska and M. Q. Zhang, "Detection of polyadenylation signals in human dna sequences.," *Gene*, vol. 231, pp. 77–86, 1999.

[31] J. H. Graber, C. R. Cantor, S. C. Mohr, and T. F. Smith, "In silico detection of control signals: Mrna 3'-end-processing sequences in diverse species.," *Proc natl acad sci u s a*, vol. 96, 1999.

[32] M. Legendre and D. Gautheret, "Sequence determinants in human polyadenylation site selection.," *Bmc genomics*, vol. 4, 2003.

[33] H. Liu, H. Han, J. Li, and L. Wong, "Dnafsminer: A web-based software toolbox to recognize two types of functional sites in dna sequences.," *Bioinformatics*, vol. 21, 2005.

[34] Y. Cheng, R. M. Miura, and B. Tian, "Prediction of mrna polyadenylation sites by support vector machine.," *Bioinformatics*, vol. 22, 2006.

[35] F. Ahmed, M. Kumar, and G. P. S. Raghava, "Prediction of polyadenylation signals in human dna sequences using nucleotide frequencies.," *In silico biol*, vol. 9, 2009.

[36] M. N. Akhtar, S. A. Bukhari, Z. Fazal, R. Qamar, and I. A. Shahmuradov, "Polyar, a new computer program for prediction of poly(a) sites in human sequences.," *Bmc genomics*, 2010.

[37] G. Ji, X. Wu, Y. Shen, J. Huang, and Q. Quinn Li, "A classification-based prediction model of messenger rna polyadenylation sites.," *J theor biol*, 2010.

[38] T.-H. Chang, L.-C. Wu, Y.-T. Chen, H.-D. Huang, B.-J. Liu, K.-F. Cheng, and J.-T. Horng, "Characterization and prediction of mrna polyadenylation sites in human genes.," *Med biol eng comput*, 2011.

[39] M. Kalkatawi, F. Rangkuti, M. Schramm, B. Jankovic, A. Kamau, R. Chowdhary, J. Archer, and V. Bajic, "Dragon polya spotter: Predictor of poly (a) motifs within human genomic dna sequences," *Bioinformatics*, vol. 28, no. 1, pp. 127–129, 2012.

[40] J. van Helden, M. del Olmo, and J. E. Prez-Ortłn, "Statistical analysis of yeast genomic downstream sequences reveals putative polyadenylation signals.," *Nucleic acids res*, 2000.

[41] D. Retelska, C. Iseli, P. Bucher, C. V. Jongeneel, and F. Naef, "Similarities and differences of polyadenylation signals in human and fly.," *Bmc genomics*, 2006.

[42] A. Lukashin and M. Borodovsky, "Genemark.hmm: New solutions for gene finding," *Nucleic acids research*, 1998.

[43] M. Stanke and S. Waack, "Gene prediction with a hidden markov model and a new intron submodel," *Bioinformatics*, 2003.

[44] C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. Noble, "Mismatch string kernels for discriminative protein classification," *Bioinformatics*, 2004.

[45] S. Sonnenburg, A. Zien, and G. Rätsch, "Arts: Accurate recognition of transcription starts in human," *Bioinformatics*, 2006.

[46] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch, "Accurate splice site prediction using support vector machines," *Bmc bioinformatics*, 2007.

[47] C. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for SVM protein classification," in *Proceedings of the pacific symposium on biocomputing*, Singapore: World Scientific Publishing, 2002, pp. 564–575.

[48] G. Rätsch and S. Sonnenburg, "Accurate splice site detection for caenorhabditis elegans," *Kernel methods in computational biology*, 2004.

[49] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *J. mach. learn. res.*, vol. 5, pp. 819–844, 2004.

[50] S. Sonnenburg, A. Zien, P. Philips, and G. Rätsch, "Poims: Positional oligomer importance matrices–understanding support vector machine-based signal detectors.," *Bioinformatics*, 2008.

[51] J. Hu, C. S. Lutz, J. Wilusz, and B. Tian, "Bioinformatic identification of candidate cis-regulatory elements involved in human mrna polyadenylation.," *Rna*, 2005.

[52] A. Parikh, L. Song, M. Ishteva, G. Teodoru, and E. Xing, "A spectral algorithm for latent junction trees," in *Conference on uncertainty in artificial intelligence*, 2012.

[53] G. Ratsch, S. Sonnenburg, and B. Scholkopf, "Rase: Recognition of alternatively spliced exons in c.elegans," *Bioinformatics*, vol. 21 Suppl 1, pp. i369–i377, 2005.

[54] D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schölkopf, "Randomized nonlinear component analysis," in *International conference on machine learning (icml)*, 2014.

[55] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Neural information processing systems*, 2009.

[56] E. Oja, "A simplified neuron model as a principal component analyzer," *J. math. biology*, vol. 15, pp. 267–273, 1982.

[57] K. Kim, M. O. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," *Ieee transactions on pattern analysis and machine intelligence*, vol. 27, no. 9, pp. 1351–1366, 2005.

[58] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," *Image processing, ieee transactions on*, vol. 16, no. 6, pp. 1662–1674, 2007.

[59] P. Honeine, "Online kernel principal component analysis: A reduced-order model," *IEEE trans. pattern anal. mach. intell.*, vol. 34, no. 9, pp. 1814–1826, 2012.

[60] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Advances in neural information processing systems*, 2014, pp. 3041–3049.

[61] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., Cambridge, MA: MIT Press, 2008.

[62] Q. Le, T. Sarlos, and A. J. Smola, "Fastfood — computing hilbert space expansions in loglinear time," in *International conference on machine learning*, 2013.

[63] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward network," *Neural networks*, vol. 2, pp. 459–473, 1989.

[64] N. N. Schraudolph, S. Günter, and S. V. N. Vishwanathan, "Fast iterative kernel PCA," in *Advances in neural information processing systems 19*, B. Schölkopf, J. Platt, and T. Hofmann, Eds., Cambridge MA: MIT Press, 2007.

[65] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental pca," in *Advances in neural information processing systems*, 2013, pp. 3174–3182.

[66] R. Arora, A. Cotter, and N. Srebro, "Stochastic optimization of pca with capped msg," in *Advances in neural information processing systems*, 2013, pp. 1815–1823.

[67] O. Shamir, "A stochastic pca algorithm with an exponential convergence rate," *Arxiv preprint arxiv:1409.2848*, 2014.

[68] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Advances in neural information processing systems*, 2014, pp. 2861–2869.

[69] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA: MIT Press, 2006.

[70] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *Aistats-12*, N. D. Lawrence and M. A. Girolami, Eds., vol. 22, 2012, pp. 583–591.

[71] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining*, ACM, 2013, pp. 239–247.

[72] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *Ieee trans. pattern anal. mach. intell.*, vol. 34, no. 3, pp. 480–492, 2012.

[73] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. W. Mahoney., "Random laplace feature maps for semigroup kernels on histograms," in *Cvpr*, 2014.

[74] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in neural information processing systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., 2009, pp. 342–350.

[75] B. Schölkopf and A. J. Smola, *Learning with kernels*. Cambridge, MA: MIT Press, 2002.

[76] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of theoretical probability*, vol. 25, no. 3, pp. 655–686, 2012.

[77] T. T. Cai and H. H. Zhou, "Optimal rates of convergence for sparse covariance matrix estimation," *The annals of statistics*, vol. 40, no. 5, pp. 2389–2420, 2012.

[78] L. Song, A. Anamdakumar, B. Dai, and B. Xie, "Nonparametric estimation of multi-view latent variable models," in *International conference on machine learning (icml)*, 2014.

[79] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of machine learning research*, vol. 3, pp. 1–48, 2002.

[80] M. Kim and V. Pavlovic, "Covariance operator based dimensionality reduction with extension to semi-supervised settings," in *International conference on artificial intelligence and statistics*, 2009, pp. 280–287.

[81] C. K. I. Williams and M. Seeger, "The effect of the input density distribution on kernel-based classifiers," in *Proc. intl. conf. machine learning*, P. Langley, Ed., San Francisco, California: Morgan Kaufmann Publishers, 2000, pp. 1159–1166.

[82] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. von Lilienfeld, and K.-R. Müller, "Learning invariant representations of molecules for atomization energy prediction.," in *Neural information processing systems*, 2012, pp. 449–457.

[83] F. Meier, P. Hennig, and S. Schaal, "Incremental local gaussian regression," in *Advances in neural information processing systems 27*, 2014, pp. 972–980.

[84] C. Boutsidis, M. Sviridenko, and D. P. Woodruff, "Optimal distributed principal component analysis," in *Manuscript*, 2015.

[85] C. Boutsidis and D. P. Woodruff, "Communication-optimal distributed principal component analysis in the column-partition model," *Corr*, vol. abs/1504.06729, 2015.

[86] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel kmeans, spectral clustering and normalized cuts," in *Conference on knowledge discovery and data mining*, 2004.

[87] M.-F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff, "Improved distributed principal component analysis," in *Advances in neural information processing systems 27*, Curran Associates, Inc., 2014, pp. 3113–3121.

[88] M.-F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," *Colt*, 2012.

[89] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," in *Advance in neural information processing systems*, 2012.

[90] R. Kannan, S. Vempala, and D. Woodruff, "Principal component analysis and higher correlations for distributed data," in *Proceedings of the 27th conference on learning theory*, 2014, pp. 1040–1057.

[91] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nyström method," *Journal of machine learning research*, vol. 13, pp. 981–1006, 2012.

[92] F. Bach and M. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the international conference on machine learning*, 2005.

[93] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Theoretical computer science*, vol. 10, no. 1-2, pp. 1–157, 2014.

[94] C. Boutsidis, D. P. Woodruff, and P. Zhong, "Communication-optimal distributed principal component analysis in the column-partition model," in *Acm symposium on theory of computing*, 2015.

[95] T. Sarlós, "Improved approximation algorithms for large matrices via random projections," in *Ieee symposium on foundations of computer science*, 2006.

[96] N. Ailon and B. Chazelle, "The fast johnson-lindenstrauss transform and approximate nearest neighbors," *SIAM journal on computing*, vol. 39, no. 1, pp. 302–322, 2009.

[97] K. L. Clarkson and D. P. Woodruff, "Low rank approximation and regression in input sparsity time," in *Proceedings of the annual acm symposium on theory of computing*, 2013.

[98] X. Meng and M. W. Mahoney, "Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression," in *Proceedings of the annual acm symposium on symposium on theory of computing*, 2013.

[99] J. Nelson and H. L. Nguyên, "Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings," in *Ieee annual symposium on foundations of computer science*, 2013.

[100] H. Avron, H. Nguyen, and D. Woodruff, "Subspace embeddings for the polynomial kernel," in *Advances in neural information processing systems*, 2014, pp. 2258–2266.

[101] P. Drineas, M. Magdon-Ismail, M. Mahoney, and D. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *The journal of machine learning research*, vol. 13, no. 1, pp. 3475–3506, 2012.

[102] P. Drineas, M. W. Mahoney, and S Muthukrishnan, "Relative-error cur matrix decompositions," *Siam journal on matrix analysis and applications*, vol. 30, no. 2, pp. 844–881, 2008.

[103] A. Deshpande and S. Vempala, "Adaptive sampling and fast low-rank matrix approximation," *Algorithms and techniques in approximation, randomization, and combinatorial optimization*, pp. 292–303, 2006.

[104] C. Boutsidis and D. P. Woodruff, "Optimal cur matrix decompositions," *Arxiv preprint arxiv:1405.7910*, 2014.

[105] K. Bache and M. Lichman, *UCI machine learning repository*, 2013.

[106] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, 2014.

[107] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *Siam review*, vol. 51, no. 4, pp. 661–703, 2009.

[108] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Nips*, 2009, pp. 342–350.

[109] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *Ieee trans. inform. theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[110] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. J. Smola, L. Song, and Z. Wang, "Deep fried convnets," *Corr*, vol. abs/1412.7149, 2014.

[111] Z. Mariet and S. Sra, "Diversity networks," *Arxiv preprint arxiv:1511.05077*, 2015.

[112] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," in *Machine learning: From theory to applications*, 1993.

[113] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks.," in *Aistats*, 2015.

[114] K. Kawaguchi, "Deep learning without poor local minima," in *Advances in neural information processing systems (nips)*, 2016.

[115] J. Lee, M. Simchowitz, M. Jordan, and B. Recht, "Gradient descent only converges to minimizers," in *Proceedings of the annual conference on learning theory (colt)*, 2016.

[116] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points – online stochastic gradient for tensor decomposition," *Arxiv preprint arxiv:1503.02101*, 2015.

[117] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in neural information processing systems*, 2014.

[118] M. Janzamin, H. Sedghi, and A. Anandkumar, "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods," *Corr abs/1506.08473*, 2015.

[119] D. Soudry and Y. Carmon, "No bad local minima: Data independent training error guarantees for multilayer neural networks," *Arxiv preprint arxiv:1605.08361*, 2016.

[120] P. Xie, Y. Deng, and E. Xing, "On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization," *Arxiv preprint arxiv:1511.07110*, 2015.

[121] E. Littwin and L. Wolf, "The multiverse loss for robust transfer learning," in *Cvpr*, 2016.

[122] D. Bilyk and M. T. Lacey, "One bit sensing, discrepancy, and stolarsky principle," *Arxiv preprint arxiv:1511.08452*, 2015.

[123] J. A. Tropp, "User-friendly tools for random matrices: An introduction," DTIC Document, Tech. Rep., 2012.

[124] M. Braun, "Accurate error bounds for the eigenvalues of the kernel matrix," *Jmlr*, vol. 7, pp. 2303–2328, 2006.

[125] T. Peel, S. Anthoine, and L. Ralaivola, "Empirical bernstein inequalities for u-statistics," in *Advances in neural information processing systems*, 2010, pp. 1903–1911.

[126] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *Journal of machine learning research*, vol. 3, pp. 463–482, 2002.

[127] D. Krotov and J. J. Hopfield, "Dense associative memory for pattern recognition," *Corr*, vol. abs/1606.01164, 2016.

[128] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge, UK: Cambridge University Press, 2004.

[129] P.-S. Huang, L. Deng, M. Hasegawa-Johnson, and X. He, "Random features for kernel deep convex network," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, IEEE, 2013, pp. 3143–3147.

[130] J. Yang, V. Sindhwani, H. Avron, and M. W. Mahoney, "Quasi-monte carlo feature maps for shift-invariant kernels," in *Proceedings of the 31th international conference on machine learning, ICML 2014, beijing, china, 21-26 june 2014*, 2014, pp. 485–493.

[131] J. Pennington, F. Yu, and S. Kumar, "Spherical random features for polynomial kernels," in *Advances in neural information processing systems*, 2015, pp. 1846–1854.

[132] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[133] A. Choromanska, Y. LeCun, and G. B. Arous, "Open problem: The landscape of the loss surfaces of multilayer networks," in *Proceedings of the 28th conference on learning theory*, 2015, pp. 1756–1760.

[134] G. Swirszcz, W. M. Czarnecki, and R. Pascanu, "Local minima in training of deep networks," *Arxiv preprint arxiv:1611.06310*, 2016.

[135] O. Shamir, "Distribution-specific hardness of learning neural networks," *Arxiv preprint arxiv:1609.01037*, 2016.

[136] B. Xie, Y. Liang, and L. Song, "Diversity leads to generalization in neural networks," *Arxiv preprint arxiv:1611.03131*, 2016.

[137] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, Jan. 1993.

[138] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. The MIT Press, 2012, ISBN: 026201825X, 9780262018258.

[139] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in neural information processing systems*, 2014, pp. 855–863.

[140] M. Hardt and T. Ma, "Identity matters in deep learning," *Arxiv preprint arxiv:1611.04231*, 2016.

[141] L. Rosasco, M. Belkin, and E. Vito, "On learning with integral operators," *Journal of machine learning research*, vol. 11, pp. 905–934, 2010.

[142] C. Müller, *Analysis of spherical symmetries in euclidean spaces*. Springer Science & Business Media, 2012, vol. 129.

[143] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," *Arxiv preprint arxiv:1011.3027*, 2010.

[144] G.-B. Huang, L. Chen, C. K. Siew, *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *Ieee trans. neural networks*, vol. 17, no. 4, pp. 879–892, 2006.

VITA

Bo Xie is a Ph.D candidate in the School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology. His primary research interest centers around solving large-scale, nonlinear and non-convex problems through scalable algorithms with theoretical guarantees. This includes large-scale kernel learning, latent variable models and deep neural networks. He has published in top-tier machine learning venues such as NIPS, ICML and KDD.